EXPERT INSIGHT

Early Access

C# 11 and .NET 7

Modern Cross-Platform Development Fundamentals

Start building websites and services with ASP.NET Core 7, Blazor, and EF Core 7

Seventh Edition

Mark J. Price

packt

EXPERT INSIGHT

Early Access

C# 11 and .NET 7

Modern Cross-Platform Development Fundamentals

Start building websites and services with ASP.NET Core 7, Blazor, and EF Core 7

Seventh Edition

Mark J. Price

packt

C# 11 and .NET 7 – Modern Cross-Platform Development Fundamentals

Copyright © 2022 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Early Access Publication: C# 11 and .NET 7 – Modern Cross-Platform Development Fundamentals

Early Access Production Reference: B18856

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK

ISBN: 978-1-80323-780-0

www.packt.com

Table of Contents

1. C# 11 and .NET 7 – Modern Cross-Platform Development Fundamentals, Seventh Edition: Start building websites and services with ASP.NET Core 7, Blazor, and EF Core 7
2. 1 Hello, C#! Welcome, .NET!
    - I. Join our book community on Discord
    - II. Setting up your development environment
        - i. Choosing the appropriate tool and application type for learning
        - ii. Deploying cross-platform
        - iii. Downloading and installing Visual Studio 2022 for Windows
        - iv. Downloading and installing Visual Studio Code
    - III. Understanding .NET
        - i. Understanding .NET Framework
        - ii. Understanding the Mono, Xamarin, and Unity projects
        - iii. Understanding .NET Core
        - iv. Understanding the journey to one .NET
        - v. Understanding .NET support
        - vi. What is different about modern .NET?
        - vii. Understanding .NET Standard
        - viii. .NET platforms and tools used by the C# and .NET book editions
        - ix. Topics covered by Apps and Services with .NET 7
        - x. Understanding intermediate language
        - xi. Comparing .NET technologies
    - IV. Building console apps using Visual Studio 2022
        - i. Managing multiple projects using Visual Studio 2022
        - ii. Writing code using Visual Studio 2022
        - iii. Compiling and running code using Visual Studio
        - iv. Understanding top-level programs
        - v. Adding a second project using Visual Studio 2022
    - V. Building console apps using Visual Studio Code
        - i. Managing multiple projects using Visual Studio Code
        - ii. Writing code using Visual Studio Code

iii. Compiling and running code using the dotnet CLI
iv. Adding a second project using Visual Studio Code
VI. Exploring code using .NET Interactive Notebooks
i. Creating a notebook
ii. Writing and running code in a notebook
iii. Saving a notebook
iv. Adding Markdown and special commands to a notebook
v. Executing code in multiple cells
vi. Using .NET Interactive Notebooks for the code in this book
VII. Reviewing the folders and files for projects
i. Understanding the common folders and files
ii. Understanding the solution code on GitHub
VIII. Making good use of the GitHub repository for this book
i. Raising issues with the book
ii. Giving me feedback
iii. Downloading solution code from the GitHub repository
iv. Using Git with Visual Studio Code and the command line
IX. Looking for help
i. Reading Microsoft documentation
ii. Getting help for the dotnet tool
iii. Getting definitions of types and their members
iv. Looking for answers on Stack Overflow
v. Searching for answers using Google
vi. Subscribing to the official .NET blog
vii. Watching Scott Hanselman's videos
viii. A companion book to continue your learning journey
X. Practicing and exploring
i. Exercise 1.1 – Test your knowledge
ii. Exercise 1.2 – Practice C# anywhere with a browser
iii. Exercise 1.3 – Explore topics
iv. Exercise 1.4 – Explore themes of modern .NET
XI. Summary
3. 02 Speaking C#
I. Join our book community on Discord
II. Introducing the C# language
i. Understanding language versions and features
ii. Understanding C# standards

    iii. Discovering your C# compiler versions
III. Understanding C# grammar and vocabulary
    i. Showing the compiler version
    ii. Understanding C# grammar
    iii. Statements
    iv. Comments
    v. Blocks
    vi. Examples of statements and blocks
    vii. Understanding C# vocabulary
    viii. Comparing programming languages to human languages
    ix. Changing the color scheme for C# syntax
    x. Help for writing correct code
    xi. Importing namespaces
    xii. Verbs are methods
    xiii. Nouns are types, variables, fields, and properties
    xiv. Revealing the extent of the C# vocabulary
IV. Working with variables
    i. Naming things and assigning values
    ii. Literal values
    iii. Storing text
    iv. Storing numbers
    v. Storing real numbers
    vi. Storing Booleans
    vii. Storing any type of object
    viii. Storing dynamic types
    ix. Declaring local variables
    x. Getting and setting the default values for types
V. Exploring more about console apps
    i. Displaying output to the user
    ii. Getting text input from the user
    iii. Simplifying the usage of the console
    iv. Getting key input from the user
    v. Passing arguments to a console app
    vi. Setting options with arguments
    vii. Handling platforms that do not support an API
VI. Understanding async and await
    i. Improving responsiveness for console apps

VII. Practicing and exploring
    i. Exercise 2.1 – Test your knowledge
    ii. Exercise 2.2 – Test your knowledge of number types
    iii. Exercise 2.3 – Practice number sizes and ranges
    iv. Exercise 2.4 – Explore topics
VIII. Summary
4. 3 Controlling Flow, Converting Types, and Handling Exceptions
I. Join our book community on Discord
II. Operating on variables
    i. Exploring unary operators
    ii. Exploring binary arithmetic operators
    iii. Assignment operators
    iv. Exploring logical operators
    v. Exploring conditional logical operators
    vi. Exploring bitwise and binary shift operators
    vii. Miscellaneous operators
III. Understanding selection statements
    i. Branching with the if statement
    ii. Pattern matching with the if statement
    iii. Branching with the switch statement
    iv. Pattern matching with the switch statement
    v. Simplifying switch statements with switch expressions
IV. Understanding iteration statements
    i. Looping with the while statement
    ii. Looping with the do statement
    iii. Looping with the for statement
    iv. Looping with the foreach statement
V. Storing multiple values in an array
    i. Working with single-dimensional arrays
    ii. Working with multi-dimensional arrays
    iii. Working with jagged arrays
    iv. Summarizing arrays
VI. Casting and converting between types
    i. Casting numbers implicitly and explicitly
    ii. Converting with the System.Convert type
    iii. Rounding numbers
    iv. Taking control of rounding rules

v. Converting from any type to a string
vi. Converting from a binary object to a string
vii. Parsing from strings to numbers or dates and times
VII. Handling exceptions
i. Wrapping error-prone code in a try block
VIII. Checking for overflow
i. Throwing overflow exceptions with the checked statement
ii. Disabling compiler overflow checks with the unchecked statement
IX. Practicing and exploring
i. Exercise 3.1 – Test your knowledge
ii. Exercise 3.2 – Explore loops and overflow
iii. Exercise 3.3 – Practice loops and operators
iv. Exercise 3.4 – Practice exception handling
v. Exercise 3.5 – Test your knowledge of operators
vi. Exercise 3.6 – Explore topics
X. Summary
5. 4 Writing, Debugging, and Testing Functions
I. Join our book community on Discord
II. Writing functions
i. Understanding top-level programs and functions
ii. Times table example
iii. Writing a function that returns a value
iv. Converting numbers from cardinal to ordinal
v. Calculating factorials with recursion
vi. Documenting functions with XML comments
vii. Using lambdas in function implementations
III. Debugging during development
i. Using Visual Studio Code integrated terminal during debugging
ii. Creating code with a deliberate bug
iii. Setting a breakpoint and start debugging
iv. Navigating with the debugging toolbar
v. Debugging windows
vi. Stepping through code
vii. Customizing breakpoints
IV. Hot reloading during development

i. Hot reloading using Visual Studio 2022
ii. Hot reloading using Visual Studio Code and the command line
V. Logging during development and runtime
i. Understanding logging options
ii. Instrumenting with Debug and Trace
iii. Configuring trace listeners
iv. Switching trace levels
VI. Unit testing
i. Understanding types of testing
ii. Creating a class library that needs testing
iii. Writing unit tests
VII. Throwing and catching exceptions in functions
i. Understanding usage errors and execution errors
ii. Commonly thrown exceptions in functions
iii. Understanding the call stack
iv. Where to catch exceptions
v. Rethrowing exceptions
vi. Implementing the tester-doer pattern
VIII. Practicing and exploring
i. Exercise 4.1 – Test your knowledge
ii. Exercise 4.2 – Practice writing functions with debugging and unit testing
iii. Exercise 4.3 – Explore topics
IX. Summary
6. 5 Building Your Own Types with Object-Oriented Programming
I. Join our book community on Discord
II. Talking about OOP
III. Building class libraries
i. Creating a class library
ii. Defining a class in a namespace
iii. Understanding members
iv. Instantiating a class
v. Importing a namespace to use a type
vi. Understanding objects
IV. Storing data within fields
i. Defining fields

    ii. Understanding access modifiers
    iii. Setting and outputting field values
    iv. Storing a value using an enum type
    v. Storing multiple values using an enum type
    vi. Storing multiple values using collections
    vii. Understanding generic collections
    viii. Making a field static
    ix. Making a field constant
    x. Making a field read-only
    xi. Initializing fields with constructors
V. Writing and calling methods
    i. Returning values from methods
    ii. Combining multiple returned values using tuples
    iii. Defining and passing parameters to methods
    iv. Overloading methods
    v. Passing optional and named parameters
    vi. Controlling how parameters are passed
    vii. Understanding ref returns
VI. Splitting classes using partial
VII. Controlling access with properties and indexers
    i. Defining read-only properties
    ii. Defining settable properties
    iii. Accessing the autogenerated private field
    iv. Requiring properties to be set during instantiation
    v. Defining indexers
VIII. More about methods
    i. Implementing functionality using methods
    ii. Implementing functionality using operators
    iii. Implementing functionality using local functions
IX. Pattern matching with objects
    i. Defining flight passengers
    ii. Enhancements to pattern matching in C# 9 or later
X. Working with records
    i. Init-only properties
    ii. Understanding records
    iii. Positional data members in records
XI. Practicing and exploring

i. Exercise 5.1 – Test your knowledge
ii. Exercise 5.2 – Explore topics
XII. Summary
7. 6 Implementing Interfaces and Inheriting Classes
I. Join our book community on Discord
II. Setting up a class library and console application
III. Making types safely reusable with generics
i. Working with non-generic types
ii. Working with generic types
IV. Raising and handling events
i. Calling methods using delegates
ii. Defining and handling delegates
iii. Defining and handling events
V. Implementing interfaces
i. Common interfaces
ii. Comparing objects when sorting
iii. Comparing objects using a separate class
iv. Implicit and explicit interface implementations
v. Defining interfaces with default implementations
VI. Managing memory with reference and value types
i. Defining reference and value types
ii. How reference and value types are stored in memory
iii. Equality of types
iv. Defining struct types
v. Defining record struct types
vi. Releasing unmanaged resources
vii. Ensuring that Dispose is called
VII. Working with null values
i. Making a value type nullable
ii. Understanding null-related initialisms
iii. Understanding nullable reference types
iv. Controlling the nullability warning check feature
v. Declaring non-nullable variables and parameters
vi. Checking for null
VIII. Inheriting from classes
i. Extending classes to add functionality
ii. Hiding members

iii. Overriding members
iv. Inheriting from abstract classes
v. Preventing inheritance and overriding
vi. Understanding polymorphism
IX. Casting within inheritance hierarchies
i. Implicit casting
ii. Explicit casting
iii. Avoiding casting exceptions
X. Inheriting and extending .NET types
i. Inheriting exceptions
ii. Extending types when you can't inherit
XI. Using an analyzer to write better code
i. Suppressing warnings
ii. Fixing the code
iii. Understanding common StyleCop recommendations
XII. Practicing and exploring
i. Exercise 6.1 – Test your knowledge
ii. Exercise 6.2 – Practice creating an inheritance hierarchy
iii. Exercise 6.3 – Explore topics
XIII. Summary
8. 7 Packaging and Distributing .NET Types
I. Join our book community on Discord
II. The road to .NET 7
i. .NET Core 1.0
ii. .NET Core 1.1
iii. .NET Core 2.0
iv. .NET Core 2.1
v. .NET Core 2.2
vi. .NET Core 3.0
vii. .NET Core 3.1
viii. .NET 5.0
ix. .NET 6.0
x. .NET 7.0
xi. Improving performance from .NET Core 2.0 to .NET 5 and later
xii. Checking your .NET SDKs for updates
III. Understanding .NET components

    i. Understanding assemblies, NuGet packages, and namespaces
    ii. Understanding the Microsoft .NET project SDKs
    iii. Understanding namespaces and types in assemblies
    iv. Understanding NuGet packages
    v. Understanding frameworks
    vi. Importing a namespace to use a type
    vii. Relating C# keywords to .NET types
    viii. Sharing code with legacy platforms using .NET Standard
    ix. Understanding defaults for class libraries with different SDKs
    x. Creating a .NET Standard 2.0 class library
    xi. Controlling the .NET SDK
IV. Publishing your code for deployment
    i. Creating a console app to publish
    ii. Understanding dotnet commands
    iii. Getting information about .NET and its environment
    iv. Managing projects
    v. Publishing a self-contained app
    vi. Publishing a single-file app
    vii. Reducing the size of apps using app trimming
V. Decompiling .NET assemblies
    i. Decompiling using the ILSpy extension for Visual Studio 2022
    ii. No, you cannot technically prevent decompilation
VI. Packaging your libraries for NuGet distribution
    i. Referencing a NuGet package
    ii. Packaging a library for NuGet
    iii. Exploring NuGet packages with a tool
    iv. Testing your class library package
VII. Porting from .NET Framework to modern .NET
    i. Could you port?
    ii. Should you port?
    iii. Differences between .NET Framework and modern .NET
    iv. Understanding the .NET Portability Analyzer
    v. Understanding the .NET Upgrade Assistant
    vi. Using non-.NET Standard libraries
VIII. Working with preview features

        i. Requiring preview features
        ii. Enabling preview features
        iii. Generic mathematics
    IX. Practicing and exploring
        i. Exercise 7.1 – Test your knowledge
        ii. Exercise 7.2 – Explore topics
        iii. Exercise 7.3 – Explore PowerShell
    X. Summary
9. 8 Working with Common .NET Types
    I. Join our book community on Discord
    II. Working with numbers
        i. Working with big integers
        ii. Working with complex numbers
        iii. Understanding quaternions
        iv. Generating random numbers for games and similar apps
    III. Working with text
        i. Getting the length of a string
        ii. Getting the characters of a string
        iii. Splitting a string
        iv. Getting part of a string
        v. Checking a string for content
        vi. Joining, formatting, and other string members
        vii. Building strings efficiently
    IV. Pattern matching with regular expressions
        i. Checking for digits entered as text
        ii. Regular expression performance improvements
        iii. Understanding the syntax of a regular expression
        iv. Examples of regular expressions
        v. Splitting a complex comma-separated string
    V. Storing multiple objects in collections
        i. Common features of all collections
        ii. Improving performance by ensuring the capacity of a collection
        iii. Understanding collection choices
        iv. Working with lists
        v. Working with dictionaries
        vi. Working with queues

vii. Sorting collections
viii. More specialized collections
ix. Working with immutable collections
x. Good practice with collections
VI. Working with spans, indexes, and ranges
i. Using memory efficiently using spans
ii. Identifying positions with the Index type
iii. Identifying ranges with the Range type
iv. Using indexes, ranges, and spans
VII. Working with network resources
i. Working with URIs, DNS, and IP addresses
ii. Pinging a server
VIII. Practicing and exploring
i. Exercise 8.1 – Test your knowledge
ii. Exercise 8.2 – Practice regular expressions
iii. Exercise 8.3 – Practice writing extension methods
iv. Exercise 8.4 – Explore topics
IX. Summary
10. 9 Working with Files, Streams, and Serialization
I. Join our book community on Discord
II. Managing the filesystem
i. Handling cross-platform environments and filesystems
ii. Managing drives
iii. Managing directories
iv. Managing files
v. Managing paths
vi. Getting file information
vii. Controlling how you work with files
III. Reading and writing with streams
i. Understanding abstract and concrete streams
ii. Writing to text streams
iii. Writing to XML streams
iv. Simplifying disposal by using the using statement
v. Compressing streams
IV. Encoding and decoding text
i. Encoding strings as byte arrays
ii. Encoding and decoding text in files

V. Reading and writing with random access handles
VI. Serializing object graphs
i. Serializing as XML
ii. Generating compact XML
iii. Deserializing XML files
iv. Serializing with JSON
v. High-performance JSON processing
vi. Controlling JSON processing
vii. New JSON extension methods for working with HTTP responses
viii. Migrating from Newtonsoft to new JSON
VII. Practicing and exploring
i. Exercise 9.1 – Test your knowledge
ii. Exercise 9.2 – Practice serializing as XML
iii. Exercise 9.3 – Explore topics
VIII. Summary
11. 10 Working with Data Using Entity Framework Core
I. Join our book community on Discord
II. Understanding modern databases
i. Understanding legacy Entity Framework
ii. Understanding Entity Framework Core
iii. Creating a console app for working with EF Core
iv. Using a sample relational database
v. Using SQLite
vi. Creating the Northwind sample database for SQLite
vii. Managing the Northwind sample database with SQLiteStudio
viii. Using the lightweight ADO.NET provider for SQLite
ix. Using Microsoft SQL Server for Windows
III. Setting up EF Core
i. Choosing an EF Core database provider
ii. Connecting to a database
iii. Defining the Northwind database context class
IV. Defining EF Core models
i. Using EF Core conventions to define the model
ii. Using EF Core annotation attributes to define the model
iii. Using the EF Core Fluent API to define the model

iv. Building EF Core models for the Northwind tables
v. Adding tables to the Northwind database context class
vi. Setting up the dotnet-ef tool
vii. Scaffolding models using an existing database
viii. Configuring preconvention models
V. Querying EF Core models
i. Filtering included entities
ii. Filtering and sorting products
iii. Getting the generated SQL
iv. Logging EF Core
v. Pattern matching with Like
vi. Generating a random number in queries
vii. Defining global filters
VI. Loading patterns with EF Core
i. Eager loading entities
ii. Enabling lazy loading
iii. Explicit loading entities
VII. Manipulating data with EF Core
i. Inserting entities
ii. Updating entities
iii. Deleting entities
iv. Pooling database contexts
VIII. Working with transactions
i. Controlling transactions using isolation levels
ii. Defining an explicit transaction
IX. Code First EF Core models
i. Understanding migrations
X. Practicing and exploring
i. Exercise 10.1 – Test your knowledge
ii. Exercise 10.2 – Practice exporting data using different serialization formats
iii. Exercise 10.3 – Explore topics
iv. Exercise 10.4 – Explore NoSQL databases
XI. Summary
12. 11 Querying and Manipulating Data Using LINQ
I. Join our book community on Discord
II. Writing LINQ expressions

        i. What makes LINQ?
        ii. Building LINQ expressions with the Enumerable class
        iii. Filtering entities with Where
        iv. Targeting a named method
        v. Simplifying the code by removing the explicit delegate instantiation
        vi. Targeting a lambda expression
        vii. Sorting entities
        viii. Declaring a query using var or a specified type
        ix. Filtering by type
        x. Working with sets and bags using LINQ
    III. Using LINQ with EF Core
        i. Building an EF Core model
        ii. Filtering and sorting sequences
        iii. Projecting sequences into new types
        iv. Joining and grouping sequences
        v. Aggregating sequences
    IV. Sweetening LINQ syntax with syntactic sugar
    V. Using multiple threads with parallel LINQ
        i. Creating an app that benefits from multiple threads
    VI. Creating your own LINQ extension methods
    VII. Working with LINQ to XML
        i. Generating XML using LINQ to XML
        ii. Reading XML using LINQ to XML
    VIII. Practicing and exploring
        i. Exercise 11.1 – Test your knowledge
        ii. Exercise 11.2 – Practice querying with LINQ
        iii. Exercise 11.3 – Explore topics
    IX. Summary
13. 12 Introducing Web Development Using ASP.NET Core
    I. Join our book community on Discord
    II. Understanding ASP.NET Core
        i. Classic ASP.NET versus modern ASP.NET Core
        ii. Building websites using ASP.NET Core
        iii. Building web and other services
    III. New features in ASP.NET Core
        i. ASP.NET Core 1.0

ii. ASP.NET Core 1.1
iii. ASP.NET Core 2.0
iv. ASP.NET Core 2.1
v. ASP.NET Core 2.2
vi. ASP.NET Core 3.0
vii. ASP.NET Core 3.1
viii. Blazor WebAssembly 3.2
ix. ASP.NET Core 5.0
x. ASP.NET Core 6.0
xi. ASP.NET Core 7.0
IV. Structuring projects
i. Structuring projects in a solution or workspace
V. Building an entity data model for the Northwind database
i. Creating a class library for entity models using SQLite
ii. Creating a class library for entity models using SQL Server
VI. Understanding web development
i. Understanding Hypertext Transfer Protocol
ii. Using Google Chrome to make HTTP requests
iii. Understanding client-side web development technologies
VII. Practicing and exploring
i. Exercise 12.1 – Explore topics
VIII. Summary
14. 13 Building Websites Using ASP.NET Core Razor Pages
I. Join our book community on Discord
II. Exploring ASP.NET Core
i. Creating an empty ASP.NET Core project
ii. Testing and securing the website
iii. Controlling the hosting environment
iv. Enabling a website to serve static content
III. Exploring ASP.NET Core Razor Pages
i. Enabling Razor Pages
ii. Adding code to a Razor Page
iii. Using shared layouts with Razor Pages
iv. Using code-behind files with Razor Pages
IV. Using Entity Framework Core with ASP.NET Core
i. Configure Entity Framework Core as a service
ii. Manipulating data using Razor Pages

iii. Injecting a dependency service into a Razor Page
V. Using Razor class libraries
i. Disabling compact folders for Visual Studio Code
ii. Creating a Razor class library
iii. Implementing a partial view to show a single employee
iv. Using and testing a Razor class library
VI. Configuring services and the HTTP request pipeline
i. Understanding endpoint routing
ii. Configuring endpoint routing
iii. Reviewing the endpoint routing configuration in our project
iv. Setting up the HTTP pipeline
v. Summarizing key middleware extension methods
vi. Visualizing the HTTP pipeline
vii. Implementing an anonymous inline delegate as middleware
VII. Enabling HTTP/3 support
VIII. Practicing and exploring
i. Exercise 13.1 – Test your knowledge
ii. Exercise 13.2 – Practice building a data-driven web page
iii. Exercise 13.3 – Practice building web pages for console apps
iv. Exercise 13.4 – Explore topics
IX. Summary
15. 14 Building Websites Using the Model-View-Controller Pattern
I. Join our book community on Discord
II. Setting up an ASP.NET Core MVC website
i. Creating an ASP.NET Core MVC website
ii. Creating the authentication database for SQL Server LocalDB
iii. Exploring the default ASP.NET Core MVC website
iv. Understanding visitor registration
v. Reviewing an MVC website project structure
vi. Reviewing the ASP.NET Core Identity database
III. Exploring an ASP.NET Core MVC website
i. Understanding ASP.NET Core MVC initialization
ii. Understanding the default MVC route
iii. Understanding controllers and actions
iv. Understanding the view search path convention
v. Understanding logging

vi. Understanding filters
vii. Understanding entity and view models
viii. Understanding views
IV. Customizing an ASP.NET Core MVC website
i. Defining a custom style
ii. Setting up the category images
iii. Understanding Razor syntax and expressions
iv. Defining a typed view
v. Passing parameters using a route value
vi. Understanding model binders in more detail
vii. Validating the model
viii. Understanding view helper methods
V. Querying a database and using display templates
VI. Improving scalability using asynchronous tasks
i. Making controller action methods asynchronous
VII. Practicing and exploring
i. Exercise 14.1 – Test your knowledge
ii. Exercise 14.2 – Practice implementing MVC by implementing a category detail page
iii. Exercise 14.3 – Practice improving scalability by understanding and implementing async action methods
iv. Exercise 14.4 – Practice unit testing MVC controllers
v. Exercise 14.5 – Explore topics
VIII. Summary
16. 15 Building and Consuming Web Services
I. Join our book community on Discord
II. Building web services using ASP.NET Core Web API
i. Understanding web service acronyms
ii. Understanding HTTP requests and responses for Web APIs
iii. Creating an ASP.NET Core Web API project
iv. Reviewing the web service's functionality
v. Creating a web service for the Northwind database
vi. Creating data repositories for entities
vii. Implementing a Web API controller
viii. Configuring the customer repository and Web API controller
ix. Specifying problem details
x. Controlling XML serialization

III. Documenting and testing web services
    i. Testing GET requests using a browser
    ii. Testing HTTP requests with the REST Client extension
    iii. Understanding Swagger
    iv. Testing requests with Swagger UI
IV. Consuming web services using HTTP clients
    i. Understanding HttpClient
    ii. Configuring HTTP clients using HttpClientFactory
    iii. Getting customers as JSON in the controller
V. Implementing advanced features for web services
    i. Implementing a Health Check API
    ii. Implementing Open API analyzers and conventions
    iii. Implementing transient fault handling
    iv. Adding security HTTP headers
    v. Enabling HTTP/3 support for HttpClient
VI. Building web services using minimal APIs
    i. Testing the minimal weather service
    ii. Adding weather forecasts to the Northwind website home page
VII. Practicing and exploring
    i. Exercise 15.1 – Test your knowledge
    ii. Exercise 15.2 – Practice creating and deleting customers with HttpClient
    iii. Exercise 15.3 – Explore topics
VIII. Summary
17. 16 Building User Interfaces Using Blazor
I. Join our book community on Discord
II. Understanding Blazor
    i. JavaScript and friends
    ii. Silverlight – C# and .NET using a plugin
    iii. WebAssembly – a target for Blazor
    iv. Understanding Blazor hosting models
    v. Understanding Blazor components
    vi. What is the difference between Blazor and Razor?
III. Comparing Blazor project templates
    i. Reviewing the Blazor Server project template
    ii. Understanding Blazor routing to page components

    iii. Running the Blazor Server project template
    iv. Reviewing the Blazor WebAssembly project template
IV. Building components using Blazor Server
    i. Defining and testing a simple Blazor Server component
    ii. Making the component a routable page component
    iii. Getting entities into a component
    iv. Abstracting a service for a Blazor component
    v. Defining forms using the EditForm component
    vi. Building a shared customer detail component
    vii. Building customer create, edit, and delete components
    viii. Testing the customer components
V. Building components using Blazor WebAssembly
    i. Configuring the server for Blazor WebAssembly
    ii. Configuring the client for Blazor WebAssembly
    iii. Testing the Blazor WebAssembly components and service
VI. Improving Blazor WebAssembly apps
    i. Enabling Blazor WebAssembly AOT
    ii. Exploring Progressive Web App support
    iii. Understanding the browser compatibility analyzer for Blazor WebAssembly
    iv. Interop with JavaScript
    v. Libraries of Blazor components
VII. Practicing and exploring
    i. Exercise 16.1 – Test your knowledge
    ii. Exercise 16.2 – Practice by creating a times table component
    iii. Exercise 16.3 – Practice by creating a country navigation item
    iv. Exercise 16.4 – Explore topics
VIII. Summary

C# 11 and .NET 7 – Modern Cross-Platform Development Fundamentals, Seventh Edition: Start building websites and services with ASP.NET Core 7, Blazor, and EF Core 7

Welcome to Packt Early Access. We're giving you an exclusive preview of this book before it goes on sale. It can take many months to write a book, but our authors have cutting-edge information to share with you today. Early Access gives you an insight into the latest developments by making chapter drafts available. The chapters may be a little rough around the edges right now, but our authors will update them over time.

You can dip in and out of this book or follow along from start to finish; Early Access is designed to be flexible. We hope you enjoy getting to know more about the process of writing a Packt book.

1. Chapter 1: Hello C#, Welcome .NET!
2. Chapter 2: Speaking C#
3. Chapter 3: Controlling Flow, Converting Types, and Handling Exceptions
4. Chapter 4: Writing, Debugging, and Testing Functions
5. Chapter 5: Building Your Own Types with Object-Oriented Programming
6. Chapter 6: Implementing Interfaces and Inheriting Classes
7. Chapter 7: Packaging and Distributing .NET Types
8. Chapter 8: Working with Common .NET Types
9. Chapter 9: Working with Files, Streams, and Serialization

10. Chapter 10: Working with Data Using Entity Framework Core
11. Chapter 11: Querying and Manipulating Data Using LINQ
12. Chapter 12: Introducing Web Development Using ASP.NET Core
13. Chapter 13: Building Websites Using ASP.NET Core Razor Pages
14. Chapter 14: Building Websites Using the Model-View-Controller Pattern
15. Chapter 15: Building and Consuming Web Services
16. Chapter 16: Building User Interface Components Using Blazor

1 Hello, C#! Welcome, .NET!

Join our book community on Discord

https://packt.link/EarlyAccess

In this first chapter, the goals are setting up your development environment, understanding the similarities and differences between modern .NET, .NET Core, .NET Framework, Mono, Xamarin, and .NET Standard, creating the simplest application possible with C# 11 and .NET 7 using various code editors, and then discovering good places to look for help.

The GitHub repository for this book has solutions using full application projects for all code tasks and notebooks when possible:

https://github.com/markjprice/cs11dotnet7

Simply press the . (dot) key or change `.com` to `.dev` in the link above to change the GitHub repository into a live code editor based on Visual Studio Code using GitHub Codespaces, as shown in *Figure 1.1*:

Figure 1.1: GitHub Codespaces live editing the book's GitHub repository

We provide you with a PDF file that has color images of the screenshots and diagrams used in this book. You can download this file from https://static.packt-cdn.com/downloads/???_ColorImages.pdf.

Visual Studio Code in a web browser is great to run alongside your chosen code editor as you work through the book's coding tasks. You can compare your code to the solution code and easily copy and paste parts if needed.

Throughout this book, I use the term **modern .NET** to refer to .NET 7 and its predecessors like .NET 5 and .NET 6 that come from .NET Core. I use the term **legacy .NET** to refer to .NET Framework, Mono, Xamarin, and .NET Standard. Modern .NET is a unification of those legacy platforms and standards.

After this first chapter, the book can be divided into three parts: first, the grammar and vocabulary of the C# language; second, the types available in .NET for building app features; and third, the fundamentals of cross-platform websites, services, and browser apps that you can build using C# and .NET.

Most people learn complex topics best by imitation and repetition rather than reading a detailed explanation of the theory; therefore, I will not overload you with detailed explanations of every step throughout this book. The idea is to get you to write some code and see it run.

You don't need to know all the nitty-gritty details immediately. That will be something that comes with time as you build your own apps and go beyond what any book can teach you.

In the words of Samuel Johnson, author of the English dictionary in 1755, I have committed "a few wild blunders, and risible absurdities, from which no work of such multiplicity is free." I take sole responsibility for these and hope you appreciate the challenge of my attempt to lash the wind by writing this book about rapidly evolving technologies like C# and .NET, and the apps that you can build with them.

If you have a complaint about this book, then please contact me before writing a negative review on Amazon. Authors cannot respond to Amazon reviews so I cannot contact you to resolve the problem and help you, or at least listen to your feedback and try to do better in the next edition. Please email me (my address is on the GitHub repository for the book) or raise an issue at the following link: https://github.com/markjprice/cs11dotnet7/issues.

This first chapter covers the following topics:

- Setting up your development environment
- Understanding .NET
- Building console apps using Visual Studio 2022
- Building console apps using Visual Studio Code
- Exploring code using .NET Interactive Notebooks
- Reviewing the folders and files for projects
- Making good use of the GitHub repository for this book
- Looking for help

Setting up your development environment

Before you start programming, you'll need a code editor for C#. Microsoft has a family of code editors and **Integrated Development Environments** (**IDEs**), which include:

- Visual Studio 2022 for Windows
- Visual Studio 2022 for Mac
- Visual Studio Code for Windows, Mac, or Linux
- Visual Studio Code for the Web
- GitHub Codespaces

Third parties have created their own C# code editors, for example, JetBrains Rider.

Choosing the appropriate tool and application type for learning

What is the best tool and application type for learning C# and .NET?

When learning, the best tool is one that helps you write code and configuration but does not hide what is really happening. IDEs provide graphical user interfaces that are friendly to use, but what are they doing for you underneath? A more basic code editor that is closer to the action while providing help to write your code is better while you are learning.

Having said that, you can make the argument that the best tool is the one you are already familiar with or that you or your team will use as your daily development tool. For that reason, I want you to be free to choose any C# code editor or IDE to complete the coding tasks in this book, including Visual Studio Code, Visual Studio for Windows, Visual Studio for Mac, or even JetBrains Rider.

In this book, I give detailed step-by-step instructions for how to create multiple projects in both Visual Studio 2022 for Windows and Visual Studio Code in *Chapter 1*. After that, I give names of projects and general instructions that work with all tools so you can use whichever tool you prefer.

The best application type for learning the C# language constructs and many of the .NET libraries is one that does not distract with unnecessary application code. For example, there is no need to create an entire Windows desktop application or a website just to learn how to write a `switch` statement.

For that reason, I believe the best method for learning the C# and .NET topics in *Chapters 1* to *11* is to build console apps. Then, in *Chapters 12* to *16*, you will build websites, services, and web browser apps.

Pros and cons of the .NET Interactive Notebooks extension

Another benefit of Visual Studio Code is the .NET Interactive Notebooks extension. This extension provides an easy and safe place to write simple code snippets for experimenting and learning. For example, data scientists use them to analyze and visualize data. Students use them to learn how to write small pieces of code for language constructs and to explore APIs.

.NET Interactive Notebooks enable you to create a single notebook file that mixes "cells" of Markdown (richly formatted text) and code using C# and other related languages, such as PowerShell, F#, and SQL (for databases).

However, .NET Interactive Notebooks do have some limitations:

- They cannot be used to create websites, services, and apps.
- They cannot read input from the user, for example, you cannot use `ReadLine` or `ReadKey`.
- They cannot have arguments passed to them.
- They do not allow you to define your own namespaces.
- They do not have any debugging tools (but these will come).

Using Visual Studio Code for cross-platform development

The most modern and lightweight code editor to choose from, and the only one from Microsoft that is cross-platform, is Visual Studio Code. It can run on all common operating systems, including Windows, macOS, and many varieties of Linux, including Red Hat Enterprise Linux (RHEL) and Ubuntu.

Visual Studio Code is a good choice for modern cross-platform development because it has an extensive and growing set of extensions to support many languages beyond C#.

Being cross-platform and lightweight, it can be installed on all platforms that your apps will be deployed to for quick bug fixes and so on. Choosing Visual Studio Code means a developer can use a cross-platform code editor to develop cross-platform apps.

Visual Studio Code has strong support for web development, although it currently has weak support for mobile and desktop development.

Visual Studio Code is supported on ARM processors so that you can develop on Apple Silicon computers and Raspberry Pi.

Visual Studio Code is by far the most popular integrated development environment, with over 70% of professional developers selecting it in the Stack Overflow 2021 survey.

Using GitHub Codespaces for development in the cloud

GitHub Codespaces is a fully configured development environment based on Visual Studio Code that can be spun up in an environment hosted in the cloud and accessed through any web browser. It supports Git repos, extensions, and a built-in command-line interface so you can edit, run, and test from any device.

Using Visual Studio for Mac for general development

Microsoft Visual Studio 2022 for Mac can create most types of applications, including console apps, websites, web services, desktop, and mobile apps.

To compile apps for Apple operating systems like iOS to run on devices like the iPhone and iPad, you must have Xcode, which only runs on macOS.

Using Visual Studio for Windows for general development

Microsoft Visual Studio 2022 for Windows can create most types of applications, including console apps, websites, web services, desktop, and mobile apps. Although you can use Visual Studio 2022 for Windows with its Xamarin extensions to write a cross-platform mobile app, you still need macOS and Xcode to compile it.

It only runs on Windows, version 7 SP1 or later. You must run it on Windows 10 or Windows 11 to create **Universal Windows Platform** (**UWP**) apps that are installed from the Microsoft Store and run in a sandbox to protect your computer.

What I used

To write and test the code for this book, I used the following hardware:

- HP Spectre (Intel) laptop
- Apple Silicon Mac mini (M1) desktop
- Raspberry Pi 400 (ARM v8) desktop

And I used the following software:

- Visual Studio Code on:
  - macOS on an Apple Silicon Mac mini (M1) desktop
  - Windows 11 on an HP Spectre (Intel) laptop
  - Ubuntu 64 on a Raspberry Pi 400
- Visual Studio 2022 for Windows on:
  - Windows 11 on an HP Spectre (Intel) laptop
- Visual Studio 2022 for Mac on:
  - macOS on an Apple Silicon Mac mini (M1) desktop

I hope that you have access to a variety of hardware and software too, because seeing the differences in platforms deepens your understanding of development challenges, although any one of the above combinations is enough to learn the fundamentals of C# and .NET and how to build practical apps and websites.

> You can learn how to write code with C# and .NET using a Raspberry Pi 400 with Ubuntu Desktop 64-bit by reading an extra article that I wrote at the following link: https://github.com/markjprice/cs9dotnet5-

extras/blob/main/raspberry-pi-ubuntu64/README.md.

Deploying cross-platform

Your choice of code editor and operating system for development does not limit where your code gets deployed.

.NET 7 supports the following platforms for deployment:

- **Windows**: Windows 7 SP1, or later. Windows 10 version 1607, or later, including Windows 11. Windows Server 2012 R2 SP1, or later. Nano Server version 1809, or later.
- **Mac**: macOS Mojave (version 10.14), or later.
- **Linux**: Alpine Linux 3.13, or later. CentOS 7, or later. Debian 10, or later. Fedora 32, or later. openSUSE 15, or later. Red Hat Enterprise Linux (RHEL) 7, or later. SUSE Enterprise Linux 12 SP2, or later. Ubuntu 16.04, 18.04, 20.04, or later.
- **Android**: API 21, or later.
- **iOS**: 10, or later.

Windows ARM64 support in .NET 5 and later means you can develop on, and deploy to, Windows ARM devices like Microsoft Surface Pro X. And developing on an Apple M1 Mac using Parallels and a Windows 11 ARM virtual machine is twice as fast.

Downloading and installing Visual Studio 2022 for Windows

Many professional Microsoft developers use Visual Studio 2022 for Windows in their day-to-day development work. Even if you choose to use Visual Studio Code to complete the coding tasks in this book, you might want to familiarize yourself with Visual Studio 2022 for Windows too. It is not until you have written a decent amount of code with a tool that you can really judge if it fits your needs.

If you do not have a Windows computer, then you can skip this section and continue to the next section where you will download and install Visual Studio Code on macOS or Linux.

Since October 2014, Microsoft has made a professional quality edition of Visual Studio for Windows available to students, open-source contributors, and individuals for free. It is called Community Edition. Any of the editions are suitable for this book. If you have not already installed it, let's do so now:

1. Download Microsoft Visual Studio 2022 version 17.1 or later for Windows from the following link: https://visualstudio.microsoft.com/downloads/.
2. Start the installer.
3. On the **Workloads** tab, select the following:
   - ASP.NET and web development
   - .NET desktop development (because this includes Console Apps)
4. On the **Individual components** tab, in the **Code tools** section, select the following:
   - Git for Windows
5. Click **Install** and wait for the installer to acquire the selected software and install it.
6. When the installation is complete, click **Launch**.
7. The first time that you run Visual Studio, you will be prompted to sign in. If you have a Microsoft account, you can use that account. If you don't, then register for a new one at the following link: https://signup.live.com/.
8. The first time that you run Visual Studio, you will be prompted to configure your environment. For **Development Settings**, choose **Visual C#**. For the color theme, I chose **Blue**, but you can choose whatever tickles your fancy.
9. If you want to customize your keyboard shortcuts, navigate to **Tools** | **Options…**, and then select the **Keyboard** section.

Microsoft Visual Studio for Windows keyboard shortcuts

In this book, I will avoid showing keyboard shortcuts since they are often customized. Where they are consistent across code editors and commonly used, I will try to show them. If you want to identify and customize your keyboard shortcuts, then you can, as shown at the following link: https://docs.microsoft.com/en-us/visualstudio/ide/identifying-and-customizing-keyboard-shortcuts-in-visual-studio.

Downloading and installing Visual Studio Code

Visual Studio Code has rapidly improved over the past couple of years and has pleasantly surprised Microsoft with its popularity. If you are brave and like to live on the bleeding edge, then there is an **Insiders** edition, which is a daily build of the next version.

Even if you plan to only use Visual Studio 2022 for Windows for development, I recommend that you download and install Visual Studio Code and try the coding tasks in this chapter using it, and then decide if you want to stick with just using Visual Studio 2022 for the rest of the book.

Let's now download and install Visual Studio Code, the .NET SDK, and the C# and .NET Interactive Notebooks extensions:

1. Download and install either the Stable build or the Insiders edition of Visual Studio Code from the following link: https://code.visualstudio.com/.

   More Information: If you need more help installing Visual Studio Code, you can read the official setup guide at the following link: https://code.visualstudio.com/docs/setup/setup-overview.

2. Download and install the .NET SDKs for versions 6.0 and 7.0 from the following link: https://www.microsoft.com/net/download.

   To fully learn how to control .NET SDKs, we need multiple versions installed. .NET 6.0 and .NET 7.0 are two currently supported versions. You can safely install multiple SDKs side by side. Although .NET 6.0 is not the most recent, it is the most recent Long Term Support (LTS) version, so that is another good reason to install it.

3. To install the C# extension, you must first launch the Visual Studio Code application.
4. In Visual Studio Code, click the **Extensions** icon or navigate to **View** | **Extensions**.
5. C# is one of the most popular extensions available, so you should see it at the top of the list, or you can enter `C#` in the search box.
6. Click **Install** and wait for supporting packages to download and install.
7. Enter `.NET Interactive` in the search box to find the **.NET Interactive Notebooks** extension.
8. Click **Install** and wait for it to install.

Installing other extensions

In later chapters of this book, you will use more Visual Studio Code extensions. If you want to install them now, all the extensions that we will use are shown in the following table:

Extension name and identifier	Description
C# for Visual Studio Code (powered by OmniSharp) `ms-dotnettools.csharp`	C# editing support, including syntax highlighting, IntelliSense, Go To Definition, Find All References, debugging support for .NET, and support for `csproj` projects on Windows, macOS, and Linux.
.NET Interactive Notebooks `ms-dotnettools.dotnet-interactive-vscode`	This extension adds support for using .NET Interactive in a Visual Studio Code notebook. It has a dependency on the Jupyter extension (`ms-toolsai.jupyter`) that itself has dependencies. Provides IntelliSense for MSBuild project files,

MSBuild project tools `tintoy.msbuild-project-tools`	including autocomplete for `<PackageReference>` elements.
REST Client `humao.rest-client`	Send an HTTP request and view the response directly in Visual Studio Code.
ilspy-vscode `icsharpcode.ilspy-vscode`	Decompile MSIL assemblies – support for modern .NET, .NET Framework, .NET Core, and .NET Standard.

Understanding Microsoft Visual Studio Code versions

Microsoft releases a new feature version of Visual Studio Code (almost) every month and bug fix versions more frequently. For example:

- Version 1.64, February 2022 feature release
- Version 1.64.1, February 2022 bug fix release

The version used in this book is 1.70, August 2022 feature release, but the version of Microsoft Visual Studio Code is less important than the version of the C# for Visual Studio Code extension that you installed.

While the C# extension is not required, it provides IntelliSense as you type, code navigation, and debugging features, so it's something that's very handy to install and keep updated to support the latest C# language features.

Microsoft Visual Studio Code keyboard shortcuts

In this book, I will avoid showing keyboard shortcuts used for tasks like creating a new file since they are often different on different operating systems. The situations where I will show keyboard shortcuts are when you need to repeatedly press the key, for example, while debugging. These are also more likely to be consistent across operating systems.

If you want to customize your keyboard shortcuts for Visual Studio Code, then you can, as shown at the following link: https://code.visualstudio.com/docs/getstarted/keybindings.

I recommend that you download a PDF of keyboard shortcuts for your operating system from the following list:

- Windows: https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf
- macOS: https://code.visualstudio.com/shortcuts/keyboard-shortcuts-macos.pdf
- Linux: https://code.visualstudio.com/shortcuts/keyboard-shortcuts-linux.pdf

Understanding .NET

.NET 7, .NET Core, .NET Framework, and Xamarin are related and overlapping platforms for developers used to build applications and services. In this section, I'm going to introduce you to each of these .NET concepts.

Understanding .NET Framework

.NET Framework is a development platform that includes a **Common Language Runtime** (**CLR**), which manages the execution of code, and a **Base Class Library** (**BCL**), which provides a rich library of classes to build applications from.

Microsoft originally designed .NET Framework to have the possibility of being cross-platform, but Microsoft put their implementation effort into making it work best with Windows.

Since .NET Framework 4.5.2, it has been an official component of the Windows operating system. Components have the same support as their parent products, so 4.5.2 and later follow the life cycle policy of the Windows OS on which they are installed. .NET Framework is installed on over one billion computers, so it must change as little as possible. Even bug fixes can cause problems, so it is updated infrequently.

For .NET Framework 4.0 or later, all the apps on a computer written for .NET Framework share the same version of the CLR and libraries stored in the **Global Assembly Cache** (**GAC**), which can lead to issues if some of them need a specific version for compatibility.

> **Good Practice**: Practically speaking, .NET Framework is Windows-only and a legacy platform. Do not create new apps using it.

Understanding the Mono, Xamarin, and Unity projects

Third parties developed a .NET Framework implementation named the **Mono** project. Mono is cross-platform, but it fell behind the official implementation of .NET Framework.

Mono has found a niche as the foundation of the **Xamarin** mobile platform as well as cross-platform game development platforms like **Unity**.

Microsoft purchased Xamarin in 2016 and now gives away what used to be an expensive Xamarin extension for free with Visual Studio. Microsoft renamed the Xamarin Studio development tool, which could only create mobile apps, to Visual Studio for Mac, and gave it the ability to create other types of projects like console apps and web services.

With Visual Studio 2022 for Mac, Microsoft has replaced parts of the Xamarin Studio editor with parts from Visual Studio 2022 for Windows to provide closer parity of experience and performance. Visual Studio 2022 for Mac was also rewritten to be a truly native macOS UI app to improve reliability and work with macOS's built-in assistive technologies.

Understanding .NET Core

Today, we live in a truly cross-platform world where modern mobile and cloud development have made Windows, as an operating system, much less important. Because of that, Microsoft has been working since 2015 on an effort to decouple .NET from its close ties with Windows. While rewriting .NET Framework to be truly cross-platform, they've taken the opportunity to refactor and remove major parts that are no longer considered core.

This new modernized product was initially branded **.NET Core** and includes a cross-platform implementation of the CLR known as **CoreCLR** and a streamlined BCL known as **CoreFX**.

Scott Hunter, Microsoft Partner Director Program Manager for .NET, has said that "Forty percent of our .NET Core customers are brand-new developers to the platform, which is what we want with .NET Core. We want to bring new people in."

.NET Core is fast-moving, and because it can be deployed side by side with an app, it can change frequently, knowing those changes will not affect other .NET Core apps on the same machine. Most improvements that Microsoft makes to .NET Core and modern .NET cannot be easily added to .NET Framework.

Understanding the journey to one .NET

At the Microsoft Build developer conference in May 2020, the .NET team announced that their plans for the unification of .NET had been delayed. They said that .NET 5 would be released on November 10, 2020, and it would unify all the various .NET platforms except mobile. It would not be until .NET 6 in November 2021 that mobile would also be supported by the unified .NET platform. Unfortunately, in September 2021 they had to

announce a six-month delay to .NET MAUI, their new cross-platform platform for mobile and desktop app development.

.NET Core has been renamed .NET and the major version number has skipped 4 to avoid confusion with .NET Framework 4.x. Microsoft plans on annual major version releases every November, rather like Apple does major version number releases of iOS every September.

The following table shows when the key versions of modern .NET were released, when future releases are planned, and which version is used by the published editions of this book:

Version	Released	Edition	Published
.NET Core RC1	November 2015	First	March 2016
.NET Core 1.0	June 2016		
.NET Core 1.1	November 2016		
.NET Core 1.0.4 and .NET Core 1.1.1	March 2017	Second	March 2017
.NET Core 2.0	August 2017		
.NET Core for UWP in Windows 10 Fall Creators Update	October 2017	Third	November 2017
.NET Core 2.1 (LTS)	May 2018		
.NET Core 2.2 (Current)	December 2018		
.NET Core 3.0 (Current)	September 2019	Fourth	October 2019
.NET Core 3.1 (LTS)	December 2019		
Blazor WebAssembly 3.2 (Current)	May 2020		
.NET 5.0 (Current)	November 2020	Fifth	November 2020
.NET 6.0 (LTS)	November 2021	Sixth	November 2021
.NET 7.0 (Current)	November 2022	Seventh	November 2022
.NET 8.0 (LTS)	November 2023	Eighth	November 2023
.NET 9.0 (Current)	November 2024	Ninth	November 2024
.NET 10.0 (LTS)	November 2025	Tenth	November 2025

Understanding Blazor WebAssembly versioning

.NET Core 3.1 included Blazor Server for building web components. Microsoft had also planned to include Blazor WebAssembly in that release, but it was delayed. Blazor WebAssembly was later released as an optional add-on for .NET Core 3.1. I include it in the table above because it was versioned as 3.2 to exclude it from the LTS of .NET Core 3.1.

Understanding .NET support

.NET versions are either **Long Term Support** (**LTS**), **Current**, or **Preview**, as described in the following list:

- **LTS** releases are stable and require fewer updates over their lifetime. These are a good choice for applications that you do not intend to update frequently. LTS releases are supported by Microsoft for 3 years after general availability, or 1 year after the next LTS release ships, whichever is longer.
- **Current** releases include features that may change based on feedback. These are a good choice for applications that you are actively developing because they provide access to the latest improvements. Current releases are supported by Microsoft for 18 months after general availability, or 6 months after the next Current or LTS release ships, whichever is longer.
- **Preview** releases are for public testing. These are a good choice for adventurous programmers who want to live on the bleeding edge, or programming book writers who need to have early access to new language features, libraries, and app platforms. Preview releases are not supported by Microsoft, but Preview or **Release Candidate (RC)** releases may be declared Go Live, meaning they are supported by Microsoft in production.

Current and LTS releases receive critical fixes throughout their lifetime for security and reliability. You must stay up to date with the latest patches to get support. For example, if a system is running 1.0 and 1.0.1 has been released, 1.0.1 will need to be installed to get support.

> **End of support** or **end of life** means the date after which bug fixes, security updates, or technical assistance are no longer available from Microsoft.

To better understand your choices of Current and LTS releases, it is helpful to see it visually, with 3-year-long black bars for LTS releases, and variable-length gray bars for Current releases that end with cross-hatching for the 6 months after a new major or minor release that they retain support for, as shown in *Figure 1.2*:

Figure 1.2: Support durations for Current and LTS releases

For example, if you had created a project using .NET 5.0, then when Microsoft released .NET 6.0 on November 8, 2021, you had to upgrade your project to .NET 6.0 by May 8, 2022 if you wanted to get fixes and updates and be supported by Microsoft.

If you need long-term support from Microsoft, then choose .NET 6.0 today and stick with it until .NET 8.0, even though Microsoft has released .NET 7.0. This is because .NET 7.0 is a Current release, and it will therefore lose support before .NET 6.0 does. Just remember that even with LTS releases you must upgrade to bug fix releases like 7.0.1 that is expected in December 2022.

At the time of publishing in November 2022, all versions of .NET Core and modern .NET have reached their end of life except those shown in the following list:

- .NET Core 3.1 will reach end of life on December 3, 2022.
- .NET 7.0 will reach end of life in early May 2024.
- .NET 6.0 will reach end of life in early November 2024.

Understanding .NET Runtime and .NET SDK versions

.NET Runtime versioning follows semantic versioning, that is, a major increment indicates breaking changes, minor increments indicate new features, and patch increments indicate bug fixes.

.NET SDK versioning does not follow semantic versioning. The major and minor version numbers are tied to the runtime version it is matched with. The patch number follows a convention that indicates the major and minor versions of the SDK.

You can see an example of this in the following table:

Change	Runtime	SDK
Initial release	7.0.0	7.0.100
SDK bug fix	7.0.0	7.0.101
Runtime and SDK bug fix	7.0.1	7.0.102
SDK new feature	7.0.1	7.0.200

Listing and removing versions of .NET

.NET Runtime updates are compatible with a major version such as 7.x, and updated releases of the .NET SDK maintain the ability to build applications that target previous versions of the runtime, which enables the safe removal of older versions.

You can see which SDKs and runtimes are currently installed using the following commands:

- `dotnet --list-sdks`
- `dotnet --list-runtimes`
- `dotnet --info`

On Windows, use the **Apps & features** section to remove .NET SDKs.

On macOS or Windows, use the `dotnet-core-uninstall` tool. This tool is not installed by default.

For example, while writing the fourth edition, I used the following command every month:

```
dotnet-core-uninstall remove --all-previews-but-latest --sdk
```

What is different about modern .NET?

Modern .NET is modularized compared to the legacy .NET Framework, which is monolithic. It is open source and Microsoft makes decisions about improvements and changes in the open. Microsoft has put particular effort into improving the performance of modern .NET.

It can be smaller than the last version of .NET Framework due to the removal of legacy and non-cross-platform technologies. For example, workloads such as **Windows Forms** and **Windows Presentation Foundation** (**WPF**) can be used to build **graphical user interface** (**GUI**) applications, but they are tightly bound to the Windows ecosystem, so they are not included with .NET on macOS and Linux.

Windows desktop development

One of the features of modern .NET is support for running old Windows Forms and WPF desktop applications using the Windows Desktop Pack that is included with the Windows version of .NET Core 3.1 or later. This is why it is bigger than the SDKs for macOS and Linux. You can make changes to your legacy Windows desktop app, and then rebuild it for modern .NET to take advantage of new features and performance improvements. I do not cover Windows desktop development in this book.

Web development

ASP.NET Web Forms and Windows Communication Foundation (WCF) are old web application and service technologies that fewer developers are choosing to use for new development projects today, so they have also been removed from modern .NET. Instead, developers prefer to use ASP.NET MVC, ASP.NET Web API, SignalR, and gRPC. These technologies have been refactored and combined into a platform that runs on modern .NET, named **ASP.NET Core**.

You'll learn about the web development technologies in *Chapter 13*, *Building Websites Using ASP.NET Core Razor Pages*, *Chapter 14*, *Building Websites Using the Model-View-Controller Pattern*, *Chapter 15*, *Building and Consuming Web Services*, and *Chapter 16*, *Building User Interfaces Using Blazor*.

> **More Information**: Some .NET Framework developers are upset that ASP.NET Web Forms, WCF, and **Windows Workflow** (**WF**) are missing from modern .NET and would like Microsoft to change their minds. There are open-source projects to enable WCF and WF to migrate to modern .NET. You can read more at the following link: https://devblogs.microsoft.com/dotnet/supporting-the-community-with-wf-and-wcf-oss-projects/. CoreWCF 1.0 was released in April 2022: https://devblogs.microsoft.com/dotnet/corewcf-v1-released/. There is an open-source project for Blazor Web Forms components at the following link: https://github.com/FritzAndFriends/BlazorWebFormsComponents.

Database development

Entity Framework (**EF**) 6 is an object-relational mapping technology that is designed to work with data that is stored in relational databases such as Oracle and Microsoft SQL Server. It has gained baggage over the years, so the cross-platform API has been slimmed down, has been given support for non-relational databases like Microsoft Azure Cosmos DB, and has been renamed **Entity Framework Core**. You will learn about it in *Chapter 10*, *Working with Data Using Entity Framework Core*.

If you have existing apps that use the old EF, then version 6.3 is supported on .NET Core 3.0 or later.

Understanding .NET Standard

The situation with .NET in 2019 was that there were three forked .NET platforms controlled by Microsoft, as shown in the following list:

- **.NET Core**: For cross-platform and new apps.
- **.NET Framework**: For legacy apps.
- **Xamarin**: For mobile apps.

Each had strengths and weaknesses because they were all designed for different scenarios. This led to the problem that a developer had to learn three platforms, each with annoying quirks and limitations.

Because of that, Microsoft defined .NET Standard – a specification for a set of APIs that all .NET platforms could implement to indicate what level of compatibility they have. For example, basic support is indicated by a platform being compliant with .NET Standard 1.4.

With .NET Standard 2.0 and later, Microsoft made all three platforms converge on a modern minimum standard, which made it much easier for developers to share code between any flavor of .NET.

For .NET Core 2.0 and later, this added most of the missing APIs that developers need to port old code written for .NET Framework to the cross-platform .NET Core. However, some APIs are implemented but throw an exception to indicate to a developer that they should not actually be used! This is usually due to differences in the operating system on which you run .NET. You'll learn how to handle these platform-specific exceptions in *Chapter 2*, *Speaking C#.*

It is important to understand that .NET Standard is just a standard. You are not able to install .NET Standard in the same way that you cannot install HTML5. To use HTML5, you must install a web browser that implements the HTML5 standard.

To use .NET Standard, you must install a .NET platform that implements the .NET Standard specification. The last .NET Standard, version 2.1, is implemented by .NET Core 3.0, Mono, and Xamarin. Some features of C# 8.0 require .NET Standard 2.1. .NET Standard 2.1 is not implemented by .NET Framework 4.8, so we should treat .NET Framework as legacy.

With the release of .NET 5 and later, the need for .NET Standard has reduced significantly because there is now a single .NET for all platforms, including mobile. Modern .NET has a single BCL and two CLRs: CoreCLR is optimized for server or desktop scenarios like websites and Windows desktop apps, and the Mono runtime is optimized for mobile and web browser apps that have limited resources.

In August 2021, Stephen Toub (Partner Software Engineer, .NET) wrote the article, "Performance Improvements in .NET 6". It has a section about Blazor and Mono where he wrote, "The runtime is itself compiled to WASM, downloaded to the browser, and used to execute the application and library code on which the app depends. I say "the runtime" here, but in reality there are actually multiple incarnations of a runtime for .NET. In .NET 6, all of the .NET core libraries for all of the .NET app models, whether it be console apps or ASP.NET Core or Blazor WASM or mobile apps, come from the same source in dotnet/runtime, but there are actually two runtime implementations in dotnet/runtime: "coreclr" and "mono"."

Read more about the two runtimes at the following link: https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-6/#blazor-and-mono

Even now, apps and websites created for .NET Framework will need to be supported, so it is important to understand that you can create .NET Standard 2.0 class libraries that are backward compatible with legacy .NET platforms.

.NET platforms and tools used by the C# and .NET book editions

For the first edition of this book, which was published in March 2016, I focused on .NET Core functionality but used .NET Framework when important or useful features had not yet been implemented in .NET Core. That was necessary because it was before the final release of .NET Core 1.0. Visual Studio 2015 was used for most examples, with Visual Studio Code shown only briefly.

The second edition was (almost) completely purged of all .NET Framework code examples so that readers were able to focus on .NET Core 1.1 examples that truly run cross-platform and it was an LTS release.

The third edition completed the switch. It was rewritten so that all the code was pure .NET Core 2.0. But giving step-by-step instructions for both Visual Studio Code and Visual Studio 2017 for all tasks added complexity.

The fourth edition continued the trend by only showing coding examples using Visual Studio Code for all but the last two chapters. In *Chapter 20*, *Building Windows Desktop Apps*, it used Visual Studio running on Windows 10, and in *Chapter 21*, *Building Cross-Platform Mobile Apps*, it used Visual Studio for Mac.

In the fifth edition, *Chapter 20*, *Building Windows Desktop Apps*, was moved to an online-only *Appendix B* to make space for a new *Chapter 20*, *Building Web User Interfaces Using Blazor*. Blazor projects can be created using Visual Studio Code.

In the sixth edition, *Chapter 19*, *Building Mobile and Desktop Apps Using .NET MAUI*, was updated to show how mobile and desktop cross-platform apps can be created using Visual Studio 2022 and **.NET MAUI** (**Multi-platform App UI**).

In this seventh edition, I refocused the book on three areas: language, libraries, and web development fundamentals. Readers can use Visual Studio Code for all examples in the book, or any other code editor of their choice.

Topics covered by Apps and Services with .NET 7

The following topics are available in a new book, *Apps and Services with .NET 7*:

Advanced libraries: Dates, times, and internationalization; reflection and Roslyn compiler; third party libraries for image handling and so on.

Data: Microsoft SQL Server, Microsoft Azure CosmosDB.

Services: gRPC, OData, GraphQL, Azure Functions, advanced Web API.

GUIs: Blazor WebAssembly, .NET MAUI.

A complete project: Services, website, and app to take and analyze surveys.

Understanding intermediate language

The C# compiler (named **Roslyn**) used by the `dotnet` CLI tool converts your C# source code into **intermediate language** (**IL**) code and stores the IL in an **assembly** (a DLL or EXE file). IL code statements are like assembly language instructions, which are executed by .NET's virtual machine, known as CoreCLR.

At runtime, CoreCLR loads the IL code from the assembly, the **just-in-time (JIT)** compiler compiles it into native CPU instructions, and then it is executed by the CPU on your machine.

The benefit of this two-step compilation process is that Microsoft can create CLRs for Linux and macOS, as well as for Windows. The same IL code runs everywhere because of the second compilation step, which generates code for the native operating system and CPU instruction set.

Regardless of which language the source code is written in, for example, C#, Visual Basic, or F#, all .NET applications use IL code for their instructions stored in an assembly. Microsoft and others provide disassembler tools that can open an assembly and reveal this IL code, such as the ILSpy .NET Decompiler extension.

Comparing .NET technologies

We can summarize and compare .NET technologies today, as shown in the following table:

Technology	Description	Host operating systems
Modern .NET	A modern feature set, full C# 8 to 11 support, used to port existing apps or create new desktop, mobile, and web apps and services. Can target older .NET platforms.	Windows, macOS, Linux, Android, iOS
.NET Framework	A legacy feature set, limited C# 8 support, no C# 9 to 11 support, used to maintain existing applications only.	Windows only
Xamarin	Mobile and desktop apps only.	Android, iOS, macOS

Building console apps using Visual Studio 2022

The goal of this section is to showcase how to build a console app using Visual Studio 2022 for Windows.

If you do not have a Windows computer or want to use Visual Studio Code, then you can skip this section since the code will be the same; just the tooling experience is different.

Managing multiple projects using Visual Studio 2022

Visual Studio 2022 has a concept named a **solution** that allows you to open and manage multiple projects simultaneously. We will use a solution to manage the two projects that you will create in this chapter.

Writing code using Visual Studio 2022

Let's get started writing code!

1. Start Visual Studio 2022.
2. In the Start window, click **Create a new project**.
3. In the **Create a new project** dialog, enter `console` in the **Search for templates** box, and select **Console App**, making sure that you have chosen the C# project template rather than another language, such as Visual Basic or C++, and that it is cross-platform, not for .NET Framework which is Windows-only, as shown in *Figure 1.3*:

Figure 1.3: Selecting the Console App project template for modern cross-platform .NET

4. Click **Next**.

5. In the **Configure your new project** dialog, enter `HelloCS` for the project name, enter `C:\cs11dotnet7` for the location, and enter `Chapter01` for the solution name, as shown in *Figure 1.4*:

Figure 1.4: Configuring names and locations for your new project

6. Click **Next**.

7. In the **Additional information** dialog, in the **Target Framework** drop-down list, note the choices of Current and long-term support versions of .NET, and then select either **.NET 6.0 (long-term support)** or **.NET 7.0 (Current)** and then click **Create**. For the projects in this chapter, we will not need any .NET 7-specific functionality.

   If you are missing .NET SDK versions, then you can install them from the following link: https://dotnet.microsoft.com/en-us/download/dotnet

8. If code is not shown, then in **Solution Explorer**, double-click to open the file named `Program.cs`, and note that **Solution Explorer** shows the **HelloCS** project, as shown in *Figure 1.5*:

Figure 1.5: Editing Program.cs in Visual Studio 2022

9. In `Program.cs`, note the code consists of only a comment and a single statement because it uses the top-level program feature introduced in C# 9, as shown in the following code:

```
// See https://aka.ms/new-console-template for more information
Console.WriteLine("Hello, World!");
```

As the comment in the code says, you can read more about this template at the following link: https://aka.ms/new-console-template

1. In `Program.cs`, modify line 2 so that the text that is being written to the console says `Hello, C#!`

All code examples and commands that the reader must read or type are shown in plain text like in *Step 9* so you will never have to read code or commands from a screenshot like in *Figure 1.5*.

Compiling and running code using Visual Studio

The next task is to compile and run the code.

1. In Visual Studio, navigate to **Debug** | **Start Without Debugging**.

2. The output in the console window will show the result of running your application, as shown in *Figure 1.6*:

Figure 1.6: Running the console app on Windows

3. Press any key to close the console app window and return to Visual Studio.
4. Optionally, you can close the **Properties** pane to make more vertical space for **Solution Explorer**.
5. Select the **HelloCS** project and note the `HelloCS.csproj` project file shows that this console app targets either `net6.0` or `net7.0`.
6. In the **Solution Explorer** toolbar, toggle on the **Show All Files** button, and note that the compiler-generated `bin` and `obj` folders are visible, as shown in *Figure 1.7*:

Figure 1.7: Showing the compiler-generated folders and files

Understanding the compiler-generated folders and files

Two compiler-generated folders were created, named `obj` and `bin`. You do not need to look inside these folders or understand their files yet (but feel free to browse around if you are curious).

Just be aware that the compiler needs to create temporary folders and files to do its work. You could delete these folders and their files, and they will be automatically recreated later. Developers often do this to "clean" a project. Visual Studio even has a command on the **Build** menu named **Clean Solution** that deletes some of these temporary files for you. The equivalent command with Visual Studio Code is `dotnet clean`.

- The `obj` folder contains one compiled *object* file for each source code file. These objects haven't been linked together into a final executable yet.
- The `bin` folder contains the *binary* executable for the application or class library. We will look at this in more detail in *Chapter 7*, *Packaging and Distributing .NET Types*.

Understanding top-level programs

If you have seen older .NET projects before then you might have expected more code, even just to output a simple message. This is because some code is written for you by the compiler when you target .NET 6 or later.

If you had created the project with .NET SDK 5.0 or earlier, the `Program.cs` file would have more statements, as shown in the following code:

```
using System;
namespace HelloCS
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Hello World!");
    }
  }
}
```

During compilation with .NET SDK 6.0 or later, all the boilerplate code to define a namespace, the `Program` class, and its `Main` method, is generated and wrapped around the statements you write. This uses a feature introduced in .NET 5 called **top-level programs**, but it was not until .NET 6 that Microsoft updated the project template for console apps to use it by default.

Key points to remember about top-level programs include the following:

- There can be only one file like this in a project.
- Any `using` statements must go at the top of the file.
- If you declare any classes or other types, they must go at the bottom of the file.
- Although you should name the method `Main` if you explicitly define it, the method is named `<Main>$` when created by the compiler.

The `using System;` statement at the top of the file imports the `System` namespace. This enables the `Console.WriteLine` statement to work. Why do we not have to import it in our project?

Implicitly imported namespaces

The trick is that we do still need to import the `System` namespace, but it is now done for us using a combination of features introduced in C# 10 and .NET 6. Let's see how:

1. In **Solution Explorer**, expand the `obj` folder, expand the `Debug` folder, expand the `net6.0` or `net7.0` folder, and open the file named `HelloCS.GlobalUsings.g.cs`.
2. Note that this file is automatically created by the compiler for projects that target .NET 6 or later, and that it uses a feature introduced in C# 10 called **global namespace imports** that imports some commonly used namespaces like `System` for use in all code files, as shown in the following code:

```
// <autogenerated />
global using global::System;
global using global::System.Collections.Generic;
global using global::System.IO;
global using global::System.Linq;
global using global::System.Net.Http;
global using global::System.Threading;
global using global::System.Threading.Tasks;
```

1. In **Solution Explorer**, click the **Show All Files** button to hide the `bin` and `obj` folders.

I will explain more about the implicit imports feature in the next chapter. For now, just note that a significant change that happened between .NET 5 and .NET 6 is that many of the project templates, like the one for console apps, use new SDK and language features to hide what is really happening.

Revealing the hidden code by throwing an exception

Now let's discover how the hidden code has been written.

1. In `Program.cs`, after the statement that outputs the message, add a statement to throw a new exception, as shown in the following code:

```
throw new Exception();
```

1. In Visual Studio, navigate to **Debug** | **Start Without Debugging**.

2. The output in the console window will show the result of running your application, including that a hidden `Program` class was defined by the compiler with a method named `<Main>$` that has a parameter named `args` for passing in arguments, as shown in *Figure 1.8*:

Figure 1.8: Throwing an exception to reveal the hidden Program.<Main>$ method

3. Press any key to close the console app window and return to Visual Studio.

Adding a second project using Visual Studio 2022

Let's add a second project to our solution to explore how to work with multiple projects:

1. In Visual Studio, navigate to **File** | **Add** | **New Project**.
2. In the **Add a new project** dialog, in **Recent project templates**, select **Console App [C#]** and then click **Next**.
3. In the **Configure your new project** dialog, for the **Project name**, enter `AboutMyEnvironment`, leave the location as `C:\cs11dotnet7\Chapter01`, and then click **Next**.

4. In the **Additional information** dialog, select **.NET 6.0 (Long-term support)** or **.NET 7.0 (Current)**, and then click **Create**.
5. In the `AboutMyEnvironment` project, in `Program.cs`, modify the statement to output the current directory and the version of the operating system, as shown in the following code:

```
Console.WriteLine(Environment.CurrentDirectory);
Console.WriteLine(Environment.OSVersion.VersionString);
```

1. In **Solution Explorer**, right-click the **Chapter01** solution, select **Set Startup Projects…**, set **Current** selection, and then click **OK**.
2. In **Solution Explorer**, click the `AboutMyEnvironment` project (or any file or folder within it), and note that Visual Studio indicates that `AboutMyEnvironment` is now the startup project by making the project name bold.

3. Navigate to **Debug** | **Start Without Debugging** to run the `AboutMyEnvironment` project, and note the result, as shown in *Figure 1.9*:

Figure 1.9: Running a top-level program in a Visual Studio solution with two projects on Windows 11

Windows 11 is just branding. Its official major version number is still 10! But its patch version is 22000 or higher.

4. Press any key to close the console app window and return to Visual Studio.

When using Visual Studio 2022 for Windows to run a console app, it executes the app from the `<projectname>\bin\Debug\net7.0` folder. It will be important to remember this when we work with the filesystem in later chapters. When using Visual Studio Code, or more accurately, the `dotnet` CLI, it has different behavior, as you are about to see.

Building console apps using Visual Studio Code

The goal of this section is to showcase how to build a console app using Visual Studio Code and the `dotnet` **command-line interface** (**CLI**).

If you never want to try Visual Studio Code or .NET Interactive Notebooks, then please feel free to skip this section and the next, and then continue with the *Reviewing the folders and files for projects* section.

Both the instructions and screenshots in this section are for Windows, but the same actions will work with Visual Studio Code on the macOS and Linux variants.

The main differences will be native command-line actions such as deleting a file: both the command and the path are likely to be different on Windows or macOS and Linux. Luckily, the `dotnet` CLI tool is identical on all platforms.

Managing multiple projects using Visual Studio Code

Visual Studio Code has a concept named a **workspace** that allows you to open and manage multiple projects simultaneously. We will use a workspace to manage the two projects that you will create in this chapter.

Writing code using Visual Studio Code

Let's get started writing code!

1. Start Visual Studio Code.
2. Make sure that you do not have any open files, folders, or workspaces.
3. Navigate to **File | Save Workspace As…**.
4. In the dialog box, navigate to your `C:` drive on Windows, your user folder on macOS (mine is named `markjprice`), or any directory or drive in which you want to save your projects.
5. Click the **New Folder** button and name the folder `cs11dotnet7`. (If you completed the section for Visual Studio 2022, then this folder will already exist.)
6. In the `cs11dotnet7` folder, create a new folder named `Chapter01-vscode`.
7. In the `Chapter01-vscode` folder, save the workspace as `Chapter01.code-workspace`.
8. Navigate to **File | Add Folder to Workspace…** or click the **Add Folder** button.
9. In the `Chapter01-vscode` folder, create a new folder named `HelloCS`.
10. Select the `HelloCS` folder and click the **Add** button.

11. When you see the dialog asking if you trust the authors of the files in this folder, click **Yes**, as shown in *Figure 1.10*:

Figure 1.10: Trusting a folder in Visual Studio Code

12. Under the menu bar, note the blue bar that explains **Restricted Mode**, and click **Manage**.

13. In the **Workspace Trust** tab, click the **Trust** button, as shown in *Figure 1.11*:

Figure 1.11: Trusting a workspace in Visual Studio Code

When working with Visual Studio Code in future, if some features seem to be broken, it might be because your workspace or folder is in restricted mode. You can scroll down the **Workspace Trust** window and manually manage which folders and workspaces you currently trust.

14. Close the **Workspace Trust** tab.
15. Navigate to **View** | **Terminal**.
16. In **TERMINAL**, make sure that you are in the `HelloCS` folder, and then use the `dotnet` CLI to create a new console app, as shown in the following command:

```
dotnet new console
```

The `dotnet new console` command targets your default .NET SDK version. To target a different version, use the `-f` switch to specify a target framework, as shown in the following command:

```
dotnet new console -f net6.0
```

1. You will see that the `dotnet` command-line tool creates a new **Console App** project for you in the current folder, and the **EXPLORER** window shows the two files created, `HelloCS.csproj` and `Program.cs`, and the `obj` folder, as shown in *Figure 1.12*:

Figure 1.12: The EXPLORER window will show that two files and a folder have been created

2. In **EXPLORER**, click on the file named `Program.cs` to open it in the editor window. The first time that you do this, Visual Studio Code may have to download and install C# dependencies like OmniSharp, .NET Core Debugger, and Razor Language Server, if it did not do this when you installed the C# extension or if they need updating. Visual Studio Code will show progress in the **Output** window and eventually the message `Finished`, as shown in the following output:

```
Installing C# dependencies...
Platform: win32, x86_64
Downloading package 'OmniSharp for Windows (.NET 4.6 / x64)' (36150 KB)..................... Done
Validating download...
Integrity Check succeeded.
Installing package 'OmniSharp for Windows (.NET 4.6 / x64)'
Downloading package '.NET Core Debugger (Windows / x64)' (45048 KB)..................... Done!
Validating download...
Integrity Check succeeded.
Installing package '.NET Core Debugger (Windows / x64)'
Downloading package 'Razor Language Server (Windows / x64)' (52344 KB)..................... Done!
Installing package 'Razor Language Server (Windows / x64)'
Finished
```

The preceding output is from Visual Studio Code on Windows. When run on macOS or Linux, the output will look slightly different, but the equivalent components for your operating system will be downloaded and installed.

1. Folders named `obj` and `bin` will have been created and when you see a notification saying that required assets are missing, click **Yes**, as shown in *Figure 1.13*:

Figure 1.13: Warning message to add required build and debug assets

If the notification disappears before you can interact with it, then you can click the bell icon in the far-right corner of the status bar to show it again.

2. After a few seconds, another folder named `.vscode` will be created with some files that are used by Visual Studio Code to provide features like IntelliSense during debugging, which you will learn more about in *Chapter 4*, *Writing, Debugging, and Testing Functions*.
3. In `Program.cs`, modify line 2 so that the text that is being written to the console says `Hello, C#!`

Good Practice: Navigate to **File** | **Auto Save**. This toggle will save the annoyance of remembering to save before rebuilding your application each time.

Compiling and running code using the dotnet CLI

The next task is to compile and run the code:

1. Navigate to **View** | **Terminal** and enter the following command:

```
dotnet run
```

1. The output in the **TERMINAL** window will show the result of running your application, as shown in *Figure 1.14*:

Figure 1.14: The output of running your first console app

2. In `Program.cs`, after the statement that outputs the message, add a statement to throw a new exception, as shown in the following code:

```
throw new Exception();
```

1. Navigate to **View** | **Terminal** and enter the following command:

```
dotnet run
```

In **TERMINAL**, you can press the up and down arrows to loop through previous commands and then press left and right arrows to edit the command before pressing Enter to run them.

1. The output in the **TERMINAL** window will show the result of running your application, including that a hidden `Program` class was defined by the compiler with a method named `<Main>$` that has a parameter named `args` for passing in arguments, as shown in the following output:

```
Hello, C#!
   at Program.<Main>$(String[] args) in C:\cs11dotnet7\Chapter01-vscode\HelloCS\Program.cs:line 3
```

Adding a second project using Visual Studio Code

Let's add a second project to our workspace to explore how to work with multiple projects:

1. In Visual Studio Code, navigate to **File** | **Add Folder to Workspace…**.
2. In the `Chapter01-vscode` folder, use the **New Folder** button to create a new folder named `AboutMyEnvironment`, select it, and click **Add**.
3. When prompted if you trust the folder, click **Yes**.
4. Navigate to **Terminal** | **New Terminal**, and in the drop-down list that appears, select **AboutMyEnvironment**. Alternatively, in **EXPLORER**, right-click the `AboutMyEnvironment` folder and then select **Open in Integrated Terminal**.
5. In **TERMINAL**, confirm that you are in the `AboutMyEnvironment` folder, and then enter the command to create a new console app, as shown in the following command:

```
dotnet new console
```

Good Practice: When using workspaces, be careful when entering commands in **TERMINAL**. Be sure that you are in the correct folder before entering potentially destructive commands! That is why I got you to create a new terminal for `AboutMyEnvironment` before issuing the command to create a new console app.

1. Navigate to **View** | **Command Palette**.
2. Enter `omni`, and then, in the drop-down list that appears, select **OmniSharp: Select Project**.

3. In the drop-down list of two projects, select the **AboutMyEnvironment** project, and when prompted, click **Yes** to add required assets to debug.

   Good Practice: To enable debugging and other useful features, like code formatting and Go To Definition, you must tell OmniSharp which project you are actively working on in Visual Studio Code. You can quickly toggle active projects by clicking the project/folder to the right of the flame icon on the left side of the status bar.

4. In **EXPLORER**, in the `AboutMyEnvironment` folder, select `Program.cs`, and then change the existing statement to output the current directory and the operating system version string, as shown in the following code:

```
Console.WriteLine(Environment.CurrentDirectory);
Console.WriteLine(Environment.OSVersion.VersionString);
```

1. In **TERMINAL**, enter the command to run a program, as shown in the following command:

```
dotnet run
```

1. Note the output in the **TERMINAL** window, as shown in *Figure 1.15*:

Figure 1.15: Running a top-level program in a Visual Studio Code workspace with two projects on Windows

When using Visual Studio Code, or more accurately, the `dotnet` CLI, to run a console app, it executes the app from the `<projectname>` folder. When using Visual Studio 2022 for Windows, it executes the app from the `<projectname>\bin\Debug\net7.0` folder. It will be important to remember this when we work with the filesystem in later chapters.

If you were to run the program on macOS Big Sur, the environment operating system would be different, as shown in the following output:

```
Unix 11.2.3
```

Exploring code using .NET Interactive Notebooks

.NET Interactive Notebooks makes writing code even easier than top-level programs. It requires Visual Studio Code, so if you did not install it and the .NET Interactive Notebooks extension earlier, please install them now.

Creating a notebook

First, we need to create a notebook:

1. In Visual Studio Code, navigate to **File** | **Close Workspace**.
2. Navigate to **View** | **Command Palette**.

3. Type `.net inter`, and then select **.NET Interactive: Create new blank notebook**, as shown in *Figure 1.16*:

Figure 1.16: Creating a new blank .NET notebook

4. When prompted to select the file extension, choose **Create as '.dib'**.

   `.dib` is an experimental file format defined by Microsoft to avoid confusion and compatibility issues with the `.ipynb` format used by Python interactive notebooks. The file extension was historically only for Jupyter notebooks that can contain an interactive (I) mix of data, Python code (PY), and output in a notebook file (NB). With .NET Interactive Notebooks, the concept has expanded to allow a mix of C#, F#, SQL, HTML, JavaScript, Markdown, and other languages. `.dib` is polyglot, meaning it supports mixed languages. Conversion between the `.dib` and `.ipynb` file formats is supported.

5. Select **C#** for the default language for code cells in the notebook.
6. If a newer version of .NET Interactive is available, you might have to wait for it to uninstall the older version and install the newer one. Navigate to **View** | **Output** and select **.NET Interactive : diagnostics** in the drop-down list. Please be patient. It can take a few minutes for the notebook to appear because it must start up a hosting environment for .NET. If nothing happens after a few minutes, then close Visual Studio Code and restart it.
7. Once the .NET Interactive Notebooks extension is downloaded and installed, the **OUTPUT** window diagnostics will show that a Kernel process has started (your process numbers will be different from the output below), as shown in the following output:

```
Extension started for VS Code Stable.
Started process 16088: dotnet tool run dotnet-interactive -- notebook-parser
Starting kernel for 'untitled:Untitled-1.dib?dotnet-interactive' using: dotnet tool run dotnet-i
Kernel for 'untitled:Untitled-1.dib?dotnet-interactive' started (16392).
```

Writing and running code in a notebook

Next, we can write code in the notebook cells:

1. The first cell should already be set to **C# (.NET Interactive)**, but if it is set to anything else, then click the language selector in the bottom-right corner of the code cell and then select **C# (.NET Interactive)** as the language mode for that cell, and note your other choices of language for a code cell, as shown in *Figure 1.17*:

Figure 1.17: Changing the language for a code cell in a .NET Interactive notebook

2. Inside the **C# (.NET Interactive)** code cell, enter a statement to output a message to the console, and note that you do not need to end the statement with a semicolon, as you normally would in a full application, as shown in the following code:

```
Console.WriteLine("Hello, .NET Interactive!")
```

1. Click the **Execute Cell** button to the left of the code cell and note the output that appears under the code cell, as shown in *Figure 1.18*:

Figure 1.18: Executing a cell in a .NET Interactive notebook and seeing the output below

Saving a notebook

Like any other file, we should save the notebook before continuing further:

1. Navigate to **File | Save As…**.
2. Change to the `Chapter01-vscode` folder and save the notebook as `Chapter01.dib`.
3. Close the `Chapter01.dib` editor tab.

Adding Markdown and special commands to a notebook

We can mix and match cells containing Markdown and code with special commands:

1. Navigate to **File | Open File…**, and select the `Chapter01.dib` file.
2. Hover your mouse above the code block and click **+ Markdown** to add a Markdown cell.
3. Type a heading level 1, as shown in the following Markdown:

```
# Chapter 1 - Hello, C#! Welcome, .NET!
Mixing *rich* **text** and code is cool!
```

1. Click the tick in the top-right corner of the cell to stop editing the cell and view the processed Markdown.

   If your cells are in the wrong order, then you can drag and drop to rearrange them.

2. Hover between the Markdown cell and the code cell and click **+ Code**.
3. Type a special command to output version information about .NET Interactive, as shown in the following code:

```
#!about
```

1. Click the **Execute Cell** button and note the output, as shown in *Figure 1.19*:

Figure 1.19: Mixing Markdown, code, and special commands in a .NET Interactive notebook

Executing code in multiple cells

When you have multiple code cells in a notebook, you must execute the preceding code cells before their context becomes available in subsequent code cells:

1. At the bottom of the notebook, add a new code cell, and then type a statement to declare a variable and assign an integer value, as shown in the following code:

```
int number = 8;
```

1. At the bottom of the notebook, add a new code cell, and then type a statement to output the `number` variable, as shown in the following code:

```
Console.WriteLine(number);
```

1. Note the second code cell does not know about the `number` variable because it was defined and assigned in another code cell, aka context, as shown in *Figure 1.20*:

Figure 1.20: The number variable does not exist in the current cell or context

2. In the first cell, click the **Execute Cell** button to declare and assign a value to the variable.
3. In the second cell, click the **Execute Cell** button to output the `number` variable, and note that this works. (Alternatively, in the first cell, you can click the **Execute Cell and Below** button in the mini-toolbar in the top-right corner of that cell.)

Good Practice: If you have related code split between two or more cells, remember to execute the preceding cells before executing the subsequent cells. At the top of the notebook, there are the following buttons: **Clear Outputs** and **Run All**. These are very handy because you can click one and then the other to ensure that all code cells are executed properly.

Using .NET Interactive Notebooks for the code in this book

Throughout the rest of the chapters, I will not give explicit instructions to use notebooks, but the GitHub repository for the book has solution notebooks when appropriate. I expect many readers will want to run my pre-created notebooks for language and library features covered in *Chapters 1* to *11*, which they want to see in action and learn about without having to write a complete application:

https://github.com/markjprice/cs11dotnet7/tree/main/notebooks

Reviewing the folders and files for projects

In this chapter, you created two projects named `HelloCS` and `AboutMyEnvironment`.

Visual Studio Code uses a workspace file to manage multiple projects. Visual Studio 2022 uses a solution file to manage multiple projects. You also created a .NET Interactive notebook.

The result is a folder structure and files that will be repeated in subsequent chapters, although with more than just two projects, as shown in *Figure 1.21*:

Figure 1.21: Folder structure and files for the two projects in this chapter

Understanding the common folders and files

Although `.code-workspace` and `.sln` files are different, the project folders and files such as `HelloCS` and `AboutMyEnvironment` are identical for Visual Studio 2022 and Visual Studio Code. This means that you can mix and match between both code editors if you like:

- In Visual Studio 2022, with a solution open, navigate to **File | Add Existing Project…** to add a project file created by another tool.
- In Visual Studio Code, with a workspace open, navigate to **File | Add Folder to Workspace…** to add a project folder created by another tool.

Good Practice: Although the source code, like the `.csproj` and `.cs` files, is identical, the `bin` and `obj` folders that are automatically generated by the compiler could have mismatched file versions that show you errors. If you want to open the same project in both Visual Studio 2022 and Visual Studio Code, delete the

temporary `bin` and `obj` folders before opening the project in the other code editor. This explains why I asked you to create a different folder for the Visual Studio Code projects in this chapter.

Understanding the solution code on GitHub

The solution code in the GitHub repository for this book includes separate folders for Visual Studio Code, Visual Studio 2022, and .NET Interactive notebook files, as shown in the following list:

- Visual Studio 2022 solutions: https://github.com/markjprice/cs11dotnet7/tree/main/vs4win
- Visual Studio Code solutions: https://github.com/markjprice/cs11dotnet7/tree/main/vscode
- .NET Interactive Notebook solutions: https://github.com/markjprice/cs11dotnet7/tree/main/notebooks

Good Practice: If you need to, return to this chapter to remind yourself how to create and manage multiple projects in the code editor of your choice. The GitHub repository has step-by-step instructions for four code editors (Visual Studio 2022 for Windows, Visual Studio Code, Visual Studio 2022 for Mac, and JetBrains Rider), along with additional screenshots: https://github.com/markjprice/cs11dotnet7/blob/main/docs/code-editors/.

Making good use of the GitHub repository for this book

Git is a commonly used source code management system. GitHub is a company, website, and desktop application that makes it easier to manage Git. Microsoft purchased GitHub in 2018, so it will continue to get closer integration with Microsoft tools.

I created a GitHub repository for this book, and I use it for the following:

- To store the solution code for the book that can be maintained after the print publication date.
- To provide extra materials that extend the book, like errata fixes, small improvements, lists of useful links, and longer articles that cannot fit in the printed book.
- To provide a place for readers to get in touch with me if they have issues with the book.

Raising issues with the book

If you get stuck following any of the instructions in this book, or if you spot a mistake in the text or the code in the solutions, please raise an issue in the GitHub repository:

1. Use your favorite browser to navigate to the following link: https://github.com/markjprice/cs11dotnet7/issues.
2. Click **New Issue**.
3. Enter as much detail as possible that will help me to diagnose the issue. For example:
    I. For a mistake in the book, the page number and section title.
    II. Your operating system, for example, Windows 11 64-bit, or macOS Big Sur version 11.2.3.
    III. Your hardware, for example, Intel, Apple Silicon, or ARM CPU.
    IV. Your code editor, for example, Visual Studio 2022, Visual Studio Code, or something else, including the version number.
    V. As much of your code and configuration that you feel is relevant and necessary.
    VI. Description of expected behavior and the behavior experienced.
    VII. Screenshots (you can drag and drop image files into the issue box).

Writing this book is a side hustle for me. I have a full-time job, so I mostly work on the book at weekends. This means that I cannot always respond immediately to issues. But I want all my readers to be successful with my book, so if I can help you (and others) without too much trouble, then I will gladly do so.

If you have a complaint about this book, then please contact me before writing a negative review on Amazon. Authors cannot respond to Amazon reviews so I cannot contact you to resolve the problem and help you, or at least listen to your feedback and try to do better in the next edition. Please email me (my address is on the

GitHub repository for the book) or raise an issue at the following link: https://github.com/markjprice/cs11dotnet7/issues.

Giving me feedback

If you'd like to give me more general feedback about the book, then either email me or the GitHub repository `README.md` page has links to some surveys. You can provide the feedback anonymously, or if you would like a response from me, then you can supply an email address. I will only use this email address to answer your feedback.

I love to hear from my readers about what they like about my book, as well as suggestions for improvements and how they are working with C# and .NET, so don't be shy. Please get in touch!

Thank you in advance for your thoughtful and constructive feedback.

Downloading solution code from the GitHub repository

I use GitHub to store solutions to all the hands-on, step-by-step coding examples throughout chapters and the practical exercises that are featured at the end of each chapter. You will find the repository at the following link: https://github.com/markjprice/cs11dotnet7.

If you just want to download all the solution files without using Git, click the green **Code** button and then select **Download ZIP**, as shown in *Figure 1.22*:

Figure 1.22: Downloading the repository as a ZIP file

I recommend that you add the preceding link to your favorite bookmarks.

Using Git with Visual Studio Code and the command line

Visual Studio Code has support for Git, but it will use your operating system's Git installation, so you must install Git 2.0 or later first before you get these features.

You can install Git from the following link: https://git-scm.com/download.

If you like to use a GUI, you can download GitHub Desktop from the following link: https://desktop.github.com.

Cloning the book solution code repository

Let's clone the book solution code repository. In the steps that follow, you will use the Visual Studio Code terminal, but you could enter the commands at any command prompt or terminal window:

1. Create a folder named `Repos-vscode` in your user or `Documents` folder, or wherever you want to store your Git repositories.

2. In Visual Studio Code, open the `Repos-vscode` folder.
3. Navigate to **View** | **Terminal**, and enter the following command:

```
git clone https://github.com/markjprice/cs11dotnet7.git
```

Note that cloning all the solutions for all the chapters will take a minute or so, so please be patient.

Looking for help

This section is all about how to find quality information about programming on the web.

Reading Microsoft documentation

The definitive resource for getting help with Microsoft developer tools and platforms is Microsoft Docs, and you can find it at the following link: https://docs.microsoft.com/.

Getting help for the dotnet tool

At the command line, you can ask the `dotnet` tool for help with its commands:

1. To open the official documentation in a browser window for the `dotnet new` command, enter the following at the command line or in the Visual Studio Code terminal:

```
dotnet help new
```

1. To get help output at the command line, use the `-h` or `--help` flag, as shown in the following command:

```
dotnet new console -h
```

1. You will see the following partial output:

```
Console App (C#)
Author: Microsoft
Description: A project for creating a command-line application that can run on .NET Core on Windo
Options:
  -f|--framework. The target framework for the project.
                      net7.0           - Target net7.0
                      net6.0           - Target net6.0
                      net5.0           - Target net5.0
                      netcoreapp3.1.   - Target netcoreapp3.1
                  Default: net7.0
  --langVersion    Sets langVersion in the created project file
                   text - Optional
  --no-restore    If specified, skips the automatic restore of the project on create.
                  bool - Optional
                  Default: false
To see help for other template languages (F#, VB), use --language option:
   dotnet new console -h --language F#
```

Getting definitions of types and their members

One of the most useful features of a code editor is **Go To Definition**. It is available in Visual Studio Code and Visual Studio 2022. It will show what the public definition of the type or member looks like by reading the metadata in the compiled assembly.

Some tools, such as ILSpy .NET Decompiler, will even reverse-engineer from the metadata and IL code back into C# or another language for you.

Let's see how to use the **Go To Definition** feature:

1. In Visual Studio 2022 or Visual Studio Code, open the solution/workspace named `Chapter01`.

2. In the `HelloCS` project, at the bottom of `Program.cs`, enter the following statement to declare an integer variable named `z`:

```
int z;
```

1. Click inside `int` and then right-click and choose **Go To Definition**.
2. In the code window that appears, you can see how the `int` data type is defined, as shown in *Figure 1.23*:

Figure 1.23: The int data type metadata

You can see that `int`:

- Is defined using the `struct` keyword
- Is in the `System.Runtime` assembly
- Is in the `System` namespace
- Is named `Int32`
- Is therefore an alias for the `System.Int32` type
- Implements interfaces such as `IComparable`
- Has constant values for its maximum and minimum values
- Has methods such as `Parse`

Good Practice: When you try to use **Go To Definition** in Visual Studio Code, you will sometimes see an error saying **No definition found**. This is because the C# extension does not know about the current project.

To fix this issue, navigate to **View** | **Command Palette**, enter `omni`, select **OmniSharp: Select Project**, and then select the project that you want to work with.

Right now, the **Go To Definition** feature is not that useful to you because you do not yet know what all of this information means.

By the end of the first part of this book, which consists of *Chapters 2* to *6*, and which teaches you about the C# language, you will know enough for this feature to become very handy.

1. In the code editor window, scroll down to find the `Parse` method with a single `string` parameter, as shown in the following code:

```
public static int Parse(string s)
```

1. Expand the code and review the comments that document this method, as shown in *Figure 1.24*:

Figure 1.24: The comments for the Parse method with a single string parameter

In the comments, you will see that Microsoft has documented the following:

- A summary that describes the method.
- Parameters like the `string` value that can be passed to the method.
- The return value of the method, including its data type.
- Three exceptions that might occur if you call this method, including `ArgumentNullException`, `FormatException`, and `OverflowException`. Now, we know that we could choose to wrap a call to this

method in a `try` statement and which exceptions to catch.

Hopefully, you are getting impatient to learn what all this means!

Be patient for a little longer. You are almost at the end of this chapter, and in the next chapter, you will dive into the details of the C# language. But first, let's see where else you can look for help.

Looking for answers on Stack Overflow

Stack Overflow is the most popular third-party website for getting answers to difficult programming questions. It's so popular that search engines such as DuckDuckGo have a special way to write a query to search the site:

1. Start your favorite web browser.
2. Navigate to `DuckDuckGo.com`, enter the following query, and note the search results, which are also shown in *Figure 1.25*:

```
!so securestring
```

Figure 1.25: Stack Overflow search results for securestring

Searching for answers using Google

You can search Google with advanced search options to increase the likelihood of finding what you need:

1. Navigate to Google.
2. Search for information about `garbage collection` using a simple Google query and note that you will probably see a lot of ads for garbage collection services in your local area before you see the Wikipedia definition of garbage collection in computer science.
3. Improve the search by restricting it to a useful site such as Stack Overflow, and by removing languages that we might not care about, such as C++, Rust, and Python, or by adding C# and .NET explicitly, as shown in the following search query:

```
garbage collection site:stackoverflow.com +C# -Java
```

Subscribing to the official .NET blog

To keep up to date with .NET, an excellent blog to subscribe to is the official .NET Blog, written by the .NET engineering teams, and you can find it at the following link: https://devblogs.microsoft.com/dotnet/.

Watching Scott Hanselman's videos

Scott Hanselman from Microsoft has an excellent YouTube channel about computer stuff they didn't teach you: http://computerstuffthey didntteachyou.com/.

I recommend it to everyone working with computers.

A companion book to continue your learning journey

I have created a second book that continues your learning journey, so it acts as a companion to this book. This first book (the one you are reading) takes a reader from beginner to intermediate level. The second book takes a reader from intermediate to advanced level.

This first book covers the fundamentals of C#, .NET, and ASP.NET Core for web development. The second book covers more specialized topics like building services with OData, GraphQL, gRPC, SignalR, and Azure Functions, as well as building graphical user interfaces for websites, desktop, and mobile apps with Blazor and .NET MAUI, as shown in *Figure 1.26*:

Figure 1.26: A companion book for learning C# and .NET

Practicing and exploring

Let's now test your knowledge and understanding by trying to answer some questions, getting some hands-on practice, and going into the topics covered throughout this chapter in greater detail.

Exercise 1.1 – Test your knowledge

Try to answer the following questions, remembering that although most answers can be found in this chapter, you should do some online research or code writing to answer others:

1. Is Visual Studio 2022 better than Visual Studio Code?
2. Is .NET 5 and later better than .NET Framework?

3. What is .NET Standard and why is it still important?
4. Why can a programmer use different languages, for example, C# and F#, to write applications that run on .NET?
5. What is a top-level program and how do you access any command-line arguments?
6. What is the name of the entry point method of a .NET console app and how should it be explicitly declared if you are not using the top-level program feature?
7. What do you type at the command-line to build and execute C# source code?
8. What are some benefits of using .NET Interactive Notebooks to write C# code?
9. Where would you look for help for a C# keyword?
10. Where would you look first for solutions to common programming problems?

Appendix, *Answers to the Test Your Knowledge Questions*, is available to download from a link in the README on the GitHub repository: https://github.com/markjprice/cs11dotnet7.

Exercise 1.2 – Practice C# anywhere with a browser

You don't need to download and install Visual Studio Code or even Visual Studio 2022 for Windows or Mac to write C#. You can start coding online at any of the following links:

- Visual Studio Code for Web: https://vscode.dev/
- C# Online Compiler | .NET Fiddle: https://dotnetfiddle.net/
- W3Schools C# Online Compiler: https://www.w3schools.com/cs/cs_compiler.php

Exercise 1.3 – Explore topics

A book is a curated experience. I have tried to find the right balance of topics to include in the printed book. Other content that I have written can be found in the GitHub repository for this book.

I believe that this book covers all the fundamental knowledge and skills a C# and .NET developer should have or be aware of. Some longer examples are best included as links to Microsoft documentation or third-party article authors.

Use the links on the following page to learn more details about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-1---hello-c-welcome-net

Exercise 1.4 – Explore themes of modern .NET

Microsoft has created a website using Blazor that shows the major themes of modern .NET:

https://themesof.net/

Summary

In this chapter, we:

- Set up your development environment.
- Discussed the similarities and differences between modern .NET, .NET Core, .NET Framework, Xamarin, and .NET Standard.
- Used Visual Studio 2022 for Windows and Visual Studio Code with the .NET SDK to create some simple console apps managed with a solution or workspace.
- Used .NET Interactive Notebooks to execute snippets of code.
- Learned how to download the solution code for this book from its GitHub repository.
- And, most importantly, learned how to find help.

In the next chapter, you will learn how to "speak" C#.

02 Speaking C#

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is all about the basics of the C# programming language. Over the course of this chapter, you'll learn how to write statements using the grammar of C#, as well as being introduced to some of the common vocabulary that you will use every day. In addition to this, by the end of the chapter, you'll feel confident in knowing how to temporarily store and work with information in your computer's memory.

This chapter covers the following topics:

- Introducing the C# language
- Understanding C# grammar and vocabulary
- Working with variables
- Exploring more about console apps
- Understanding async and await

Introducing the C# language

This part of the book is about the C# language—the grammar and vocabulary that you will use every day to write the source code for your applications.

Programming languages have many similarities to human languages, except that in programming languages, you can make up your own words, just like Dr. Seuss!

In a book written by Dr. Seuss in 1950, *If I Ran the Zoo*, he states this:

> *"And then, just to show them, I'll sail to Ka-Troo And Bring Back an It-Kutch, a Preep, and a Proo, A Nerkle, a Nerd, and a Seersucker, too!"*

Understanding language versions and features

This part of the book covers the C# programming language and is written primarily for beginners, so it covers the fundamental topics that all developers need to know, from declaring variables to storing data to how to define your own custom data types.

This book covers features of the C# language from version 1 up to the latest version 11.

If you already have some familiarity with older versions of C# and are excited to find out about the new features in the most recent versions of C#, I have made it easier for you to jump around by listing language versions and their important new features below, along with the chapter number and topic title where you can learn about them.

Project COOL

Before the first release of C#, it had the codename COOL (C-like Object Oriented Language).

C# 1

C# 1 was released in February 2002 and included all the important features of a statically typed object-oriented modern language, as you will see throughout *Chapters 2* to *6*.

C# 1.2

C# 1.2, with a few minor improvements like automatic disposal at the end of `foreach` statements, was released with Visual Studio .NET 2003.

C# 2

C# 2 was released in 2005 and focused on enabling strong data typing using generics, to improve code performance and reduce type errors, including the topics listed in the following table:

Feature	Chapter	Topic
Nullable value types	6	Making a value type nullable
Generics	6	Making types more reusable with generics

C# 3

C# 3 was released in 2007 and focused on enabling declarative coding with **Language INtegrated Queries** (**LINQ**) and related features like anonymous types and lambda expressions, including the topics listed in the following table:

Feature	Chapter	Topic
Implicitly typed local variables	2	Inferring the type of a local variable
LINQ	11	All topics in *Chapter 11* , *Querying and Manipulating Data Using LINQ*

C# 4

C# 4 was released in 2010 and focused on improving interoperability with dynamic languages like F# and Python, including the topics listed in the following table:

Feature	Chapter	Topic
Dynamic types	2	Storing dynamic types
Named/optional arguments	5	Optional parameters and named arguments

C# 5

C# 5 was released in 2012 and focused on simplifying asynchronous operation support by automatically implementing complex state machines while writing what looks like synchronous statements, including the topics listed in the following table:

Feature	Chapter	Topic
Simplified asynchronous tasks	2	Understanding async and await

C# 6

C# 6 was released in 2015 and focused on minor refinements to the language, including the topics listed in the following table:

Feature	Chapter	Topic
`static` imports	2	Simplifying the usage of the console
Interpolated strings	2	Displaying output to the user
Expression bodied members	5	Defining read-only properties

C# 7.0

C# 7.0 was released in March 2017 and focused on adding functional language features like tuples and pattern matching, as well as minor refinements to the language, including the topics listed in the following table:

Feature	Chapter	Topic
Binary literals and digit separators	2	Storing whole numbers
Pattern matching	3	Pattern matching with the `if` statement
`out` variables	5	Controlling how parameters are passed
Tuples	5	Combining multiple values with tuples
Local functions	6	Defining local functions

C# 7.1

C# 7.1 was released in August 2017 and focused on minor refinements to the language, including the topics listed in the following table:

Feature	Chapter	Topic
`async` Main	2	Improving responsiveness for console apps
Default literal expressions	5	Setting fields with default literal
Inferred tuple element names	5	Inferring tuple names

C# 7.2

C# 7.2 was released in November 2017 and focused on minor refinements to the language, including the topics listed in the following table:

Feature	Chapter	Topic
Leading underscores in numeric literals	2	Storing whole numbers
Non-trailing named arguments	5	Optional parameters and named arguments
`private protected` access modifier	5	Understanding access modifiers
You can test `==` and `!=` with tuple types	5	Comparing tuples

C# 7.3

C# 7.3 was released in May 2018 and focused on performance-oriented safe code that improves `ref` variables, pointers, and `stackalloc`. These are advanced and rarely needed for most developers, so they are not covered in this book.

C# 8

C# 8 was released in September 2019 and focused on a major change to the language related to null handling, including the topics listed in the following table:

Feature	Chapter	Topic
Switch expressions	3	Simplifying `switch` statements with switch expressions

Nullable reference types	6	Making a reference type nullable
Default interface methods	6	Understanding default interface methods

C# 9

C# 9 was released in November 2020 and focused on record types, refinements to pattern matching, and minimal-code projects, including the topics listed in the following table:

Feature	Chapter	Topic
Minimal-code console apps	1	Top-level programs
Target-typed new	2	Using target-typed new to instantiate objects
Enhanced pattern matching	5	Pattern matching with objects
Records	5	Working with records

C# 10

C# 10 was released in November 2021 and focused on features that minimize the amount of code needed in common scenarios, including the topics listed in the following table:

Feature	Chapter	Topic
Global namespace imports	2	Importing namespaces
Constant string literals	2	Formatting using interpolated strings
File-scoped namespaces	5	Simplifying namespace declarations
Record structs	6	Working with record struct types
`ArgumentNullException.ThrowIfNull`	6	Checking for null in method parameters

C# 11

C# 11 was released in November 2022 and focused on features that simplify your code, including the topics listed in the following table:

Feature	Chapter	Topic
Raw string literals	2	Understanding raw string literals
Required properties	5	Requiring properties to be set during instantiation
`!!` null parameter check operator	6	Checking for null in method parameters

Early Access Program: To use C# 11 features with .NET SDK 6.0.200 or later, or with .NET 7 previews, you must add the following element to your project file: `<LangVersion>preview</LangVersion>`

Understanding C# standards

Over the years, Microsoft has submitted a few versions of C# to standards bodies, as shown in the following table:

C# version	ECMA standard	ISO/IEC standard
1.0	ECMA-334:2003	ISO/IEC 23270:2003
2.0	ECMA-334:2006	ISO/IEC 23270:2006
5.0	ECMA-334:2017	ISO/IEC 23270:2018

The ECMA standard for C# 6 is still a draft and work on adding C# 7 features is progressing. Microsoft made C# open source in 2014. You can read the C# ECMA standard document at the following link: https://www.ecma-international.org/publications-and-standards/standards/ecma-334/

More practically useful than the ECMA standards are the public GitHub repositories for making the work on C# and related technologies as open as possible, as shown in the following table:

Description	Link
C# language design	https://github.com/dotnet/csharplang
Compiler implementation	https://github.com/dotnet/roslyn
Standard to describe the language	https://github.com/dotnet/csharpstandard

Discovering your C# compiler versions

The .NET language compiler for C# and Visual Basic, also known as **Roslyn**, along with a separate compiler for F#, is distributed as part of the .NET SDK. To use a specific version of C#, you must have at least that version of the .NET SDK installed, as shown in the following table:

.NET SDK	Roslyn compiler	Default C# language
1.0.4	2.0 - 2.2	7.0
1.1.4	2.3 - 2.4	7.1
2.1.2	2.6 - 2.7	7.2
2.1.200	2.8 - 2.10	7.3
3.0	3.0 - 3.4	8.0
5.0	3.8	9.0
6.0	4.0	10.0
7.0	?	11:0

When you create class libraries, you can choose to target .NET Standard as well as versions of modern .NET. They have default C# language versions, as shown in the following table:

.NET Standard	C#
2.0	7.3
2.1	8.0

> Although you must have a minimum version of the .NET SDK installed to have access to a specific compiler version, the projects that you create can target older versions of .NET and still use a modern compiler version. For example, if you have the .NET 7 SDK or later installed, then you could use C# 11 language features in a console app that targets .NET Core 3.0.

How to output the SDK version

Let's see what .NET SDK and C# language compiler versions you have available:

1. On Windows, start **Windows Terminal** or **Command Prompt**. On macOS, start **Terminal**.
2. To determine which version of the .NET SDK you have available, enter the following command:

```
dotnet --version
```

1. Note the version at the time of publishing is 7.0.100, indicating that it is the initial version of the SDK without any bug fixes or new features yet, as shown in the following output:

```
7.0.100
```

Enabling a specific language version compiler

Developer tools like Visual Studio and the `dotnet` command-line interface assume that you want to use the latest major version of a C# language compiler by default. Before C# 8.0 was released, C# 7.0 was the latest major

version and was used by default. To use the improvements in a C# point release like 7.1, 7.2, or 7.3, you had to add a `<LangVersion>` configuration element to the project file, as shown in the following markup:

```
<LangVersion>7.3</LangVersion>
```

After the release of C# 11 with .NET 7, if Microsoft releases a C# 11.1 compiler and you want to use its new language features, then you will have to add a configuration element to your project file, as shown in the following markup:

```
<LangVersion>11.1</LangVersion>
```

Potential values for the `<LangVersion>` are shown in the following table:

LangVersion	Description
`7` , `7.1` , `7.2` , `7.3` , `8` , `9` , `10` , `11`	Entering a specific version number will use that compiler if it has been installed.
`latestmajor`	Uses the highest major number, for example, 7.0 in August 2019, 8 in October 2019, 9 in November 2020, 10 in November 2021, 11 in November 2022.
`latest`	Uses the highest major and highest minor number, for example, 7.2 in 2017, 7.3 in 2018, 8 in 2019, perhaps 11.1 in H1 2023.
`preview`	Uses the highest available preview version, for example, 11.0 in July 2022 with .NET 7.0 Preview 6 installed.

After creating a new project, you can edit the `.csproj` file and add the `<LangVersion>` element, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net7.0</TargetFramework>
    <LangVersion>preview</LangVersion>
  </PropertyGroup>
</Project>
```

C# 11 is the first compiler supported by an older version of the .NET SDK as well as the latest. My guess is that this is because .NET 6 is an LTS release so Microsoft knows developers will want to continue to use .NET 6 longer than .NET 7, so they have decided to give both versions the ability to use C# 11.

With .NET SDK 6.0.200 and later released in February 2022, you could set the language version to `preview` to start exploring C# 11 language features. I expect that whatever version is released alongside .NET 7 GA in November 2022, it will default to using the C# 10 compiler unless you explicitly set the language version to 11, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net6.0</TargetFramework>
    <LangVersion>11</LangVersion>
  </PropertyGroup>
</Project>
```

If you target `net7.0`, then the default language will be C# 11.

> **Good Practice**: If you are using Visual Studio Code and you have not done so already, install the Visual Studio Code extension named **MSBuild project tools**. This will give you IntelliSense while editing `.csproj` files, including making it easy to add the `<LangVersion>` element with appropriate values.

Understanding C# grammar and vocabulary

To learn simple C# language features, you can use .NET Interactive Notebooks, which remove the need to create an application of any kind.

To learn some other C# language features, you will need to create an application. The simplest type of application is a console app.

Let's start by looking at the basics of the grammar and vocabulary of C#. Throughout this chapter, you will create multiple console apps, with each one showing related features of the C# language.

Showing the compiler version

We will start by writing code that shows the compiler version:

1. If you've completed *Chapter 1*, *Hello, C#! Welcome, .NET!*, then you will already have a `cs11dotnet7` folder. If not, then you'll need to create it.

2. Use your preferred code editor to create a new project, as defined in the following list:

   - Project template: **Console App [C#]** / `console`
   - Project file and folder: `Vocabulary`
   - Workspace/solution file and folder: `Chapter02`

   Good Practice: If you have forgotten how, or did not complete the previous chapter, then step-by-step instructions for creating a workspace/solution with multiple projects are given in *Chapter 1*, *Hello, C#! Welcome, .NET!*.

3. Open the `Program.cs` file, and under the comment, add a statement to show the C# version as an error, as shown in the following code:

```
#error version
```

1. Run the console app:
   - If you are using Visual Studio Code, then in a terminal, enter the command `dotnet run`.
   - If you are using Visual Studio 2022, then navigate to **Debug | Start Without Debugging**. When prompted to continue and run the last successful build, click **No**.

2. Note the compiler version and language version appear as a compiler error message number `CS8304`, as shown in *Figure 2.1*:

Figure 2.1: A compiler error that shows the C# language version

3. The error message in the Visual Studio Code **PROBLEMS** window or Visual Studio **Error List** window says `Compiler version: '5.0.0...'` with language version `11.0`.

4. Comment out the statement that causes the error, as shown in the following code:

```
// #error version
```

1. Note that the compiler error messages disappear.

Understanding C# grammar

The grammar of C# includes statements and blocks. To document your code, you can use comments.

> **Good Practice**: Comments should not be the only way that you document your code. Choosing sensible names for variables and functions, writing unit tests, and creating actual documents are other ways to document your code.

Statements

In English, we indicate the end of a sentence with a full stop. A sentence can be composed of multiple words and phrases, with the order of words being part of the grammar. For example, in English, we say "the black cat."

The adjective, *black*, comes before the noun, *cat*. Whereas French grammar has a different order; the adjective comes after the noun: "le chat noir." What's important to take away from this is that the order matters.

C# indicates the end of a **statement** with a semicolon. A statement can be composed of multiple **variables** and **expressions**. For example, in the following statement, `totalPrice` is a variable and `subtotal + salesTax` is an expression:

```
var totalPrice = subtotal + salesTax;
```

The expression is made up of an operand named `subtotal`, an operator `+`, and another operand named `salesTax`. The order of operands and operators matters.

Comments

Comments are the primary method of documenting your code to increase understanding of how it works, for other developers to read or even yourself when you come back to it months later.

> In *Chapter 4, Writing, Debugging, and Testing Functions*, you will learn about XML comments that work with a tool to generate web pages to document your code.

You can add comments to explain your code using a double slash, `//`. The compiler will ignore everything after the `//` until the end of the line, as shown in the following code:

```
// sales tax must be added to the subtotal
var totalPrice = subtotal + salesTax;
```

To write a multiline comment, use `/*` at the beginning and `*/` at the end of the comment, as shown in the following code:

```
/*
This is a
multi-line comment.
*/
```

Although `/* */` is most commonly used for multi-line comments, it is also useful for commenting in the middle of a statement, as shown in the following code:

```
var totalPrice = subtotal /* for this item */ + salesTax;
```

Good Practice: Well-designed code, including function signatures with well-named parameters and class encapsulation, can be somewhat self-documenting. When you find yourself putting too many comments and explanations in your code, ask yourself: can I rewrite, aka refactor, this code to make it more understandable without long comments?

Your code editor has commands to make it easier to add and remove comment characters, as shown in the following list:

- Visual Studio 2022 for Windows: Navigate to **Edit** | **Advanced** | **Comment Selection** or **Uncomment Selection**.
- Visual Studio Code: Navigate to **Edit** | **Toggle Line Comment** or **Toggle Block Comment**.

Good Practice: You **comment** code by adding descriptive text above or after code statements. You **comment out** code by adding comment characters before or around statements to make them inactive. **Uncommenting** means removing the comment characters.

Blocks

In English, we indicate a new paragraph by starting a new line. C# indicates a **block** of code with the use of curly brackets, `{ }`.

Blocks start with a declaration to indicate what is being defined. For example, a block can define the start and end of many language constructs including namespaces, classes, methods, or statements like `foreach`.

You will learn more about namespaces, classes, and methods later in this chapter and subsequent chapters, but to briefly introduce some of those concepts now:

- A **namespace** contains types like classes to group them together.
- A **class** contains the members of an object, including methods.
- A **method** contains statements that implement an action that an object can take.

Examples of statements and blocks

In a simple console app that does not use the top-level program feature, I've added some comments to the statements and blocks, as shown in the following code:

```
using System; // a semicolon indicates the end of a statement
namespace Basics
{ // an open brace indicates the start of a block
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Hello World!"); // a statement
    }
  }
} // a close brace indicates the end of a block
```

Understanding C# vocabulary

The C# vocabulary is made up of **keywords**, **symbol characters**, and **types**.

Some of the predefined, reserved keywords that you will see in this book include `using`, `namespace`, `class`, `static`, `int`, `string`, `double`, `bool`, `if`, `switch`, `break`, `while`, `do`, `for`, `foreach`, `and`, `or`, `not`, `record`, and `init`.

Some of the symbol characters that you will see include `"`, `'`, `+`, `-`, `*`, `/`, `%`, `@`, and `$`.

There are other contextual keywords that only have a special meaning in a specific context.

However, that still means that there are only about 100 actual C# keywords in the language.

Comparing programming languages to human languages

The English language has more than 250,000 distinct words, so how does C# get away with only having about 100 keywords? Moreover, why is C# so difficult to learn if it has only 0.0416% of the number of words in the English language?

One of the key differences between a human language and a programming language is that developers need to be able to define the new "words" with new meanings. Apart from the about 100 keywords in the C# language, this book will teach you about some of the hundreds of thousands of "words" that other developers have defined, but you will also learn how to define your own "words."

Programmers all over the world must learn English because most programming languages use English words such as "namespace" and "class". There are programming languages that use other human languages, such as Arabic, but they are rare. If you are interested in learning more, this YouTube video shows a demonstration of an Arabic programming language: https://youtu.be/dkO8cdwf6v8.

Changing the color scheme for C# syntax

By default, Visual Studio 2022 and Visual Studio Code show C# keywords in blue to make them easier to differentiate from other code. Both tools allow you to customize the color scheme.

In Visual Studio 2022:

1. Navigate to **Tools** | **Options**.
2. In the **Options** dialog box, in the **Environment** section, select **Fonts and Colors**, and then select the display items that you would like to customize. You can also search for the section instead of browsing for it.

In Visual Studio Code:

1. Navigate to **File** | **Preferences** | **Color Theme**. It is in the **Code** menu on macOS.
2. Select a color theme. For reference, I'll use the **Light+ (default light)** color theme so that the screenshots look better in a printed book.

Help for writing correct code

Plain text editors such as Notepad don't help you write correct English. Likewise, Notepad won't help you write correct C# either.

Microsoft Word can help you write English by highlighting spelling mistakes with red squiggles, with Word saying that "icecream" should be ice-cream or ice cream, and grammatical errors with blue squiggles, such as a sentence should have an uppercase first letter.

Similarly, Visual Studio 2022 and Visual Studio Code's C# extension help you write C# code by highlighting spelling mistakes, such as the method name should be `WriteLine` with an uppercase `L`, and grammatical errors, such as statements that must end with a semicolon.

The C# extension constantly watches what you type and gives you feedback by highlighting problems with colored squiggly lines, like that of Microsoft Word.

Let's see it in action:

1. In `Program.cs`, change the `L` in the `WriteLine` method to lowercase.
2. Delete the semicolon at the end of the statement.

3. In Visual Studio Code, navigate to **View** | **Problems**, or in Visual Studio navigate to **View** | **Error List**, and note that a red squiggle appears under the code mistakes and details are shown, as you can see in *Figure 2.2*:

Figure 2.2: The Error List window showing two compile errors

4. Fix the two coding errors.

Importing namespaces

`System` is a namespace, which is like an address for a type. To refer to someone's location exactly, you might use `Oxford.HighStreet.BobSmith`, which tells us to look for a person named Bob Smith on the High Street in the city of Oxford.

`System.Console.WriteLine` tells the compiler to look for a method named `WriteLine` in a type named `Console` in a namespace named `System`.

To simplify our code, the **Console App** project template for every version of .NET before 6.0 added a statement at the top of the code file to tell the compiler to always look in the `System` namespace for types that haven't been prefixed with their namespace, as shown in the following code:

```
using System; // import the System namespace
```

We call this *importing the namespace*. The effect of importing a namespace is that all available types in that namespace will be available to your program without needing to enter the namespace prefix and will be seen in IntelliSense while you write code.

.NET Interactive notebooks have most namespaces imported automatically.

Implicitly and globally importing namespaces

Traditionally, every `.cs` file that needs to import namespaces would have to start with `using` statements to import those namespaces. Namespaces like `System` and `System.Linq` are needed in almost all `.cs` files, so the first few lines of every `.cs` file often had at least a few `using` statements, as shown in the following code:

```
using System;
using System.Linq;
using System.Collections.Generic;
```

When creating websites and services using ASP.NET Core, there are often dozens of namespaces that each file would have to import.

C# 10 introduced a new keyword combination and .NET SDK 6 introduced a new project setting that work together to simplify importing common namespaces.

The `global using` keyword combination means you only need to import a namespace in one `.cs` file and it will be available throughout all `.cs` files. You could put `global using` statements in the `Program.cs` file, but I recommend creating a separate file for those statements named something like `GlobalUsings.cs` with the contents being all your `global using` statements, as shown in the following code:

```
global using System;
global using System.Linq;
global using System.Collections.Generic;
```

> **Good Practice**: As developers get used to this new C# feature, I expect one naming convention for this file to become the standard. As you are about to see, the related .NET SDK feature uses a similar naming convention.

Any projects that target .NET 6.0 or later and therefore use the C# 10 or later compiler, generate a `<ProjectName>.GlobalUsings.g.cs` file in the `obj\Debug\net7.0` folder to implicitly globally import some common namespaces like `System`. The specific list of implicitly imported namespaces depends on which SDK you target, as shown in the following table:

SDK	Implicitly imported namespaces

`Microsoft.NET.Sdk`	`System`
	`System.Collections.Generic`
	`System.IO`
	`System.Linq`
	`System.Net.Http`
	`System.Threading`
	`System.Threading.Tasks`

`Microsoft.NET.Sdk.Web`	Same as `Microsoft.NET.Sdk` and:
	`System.Net.Http.Json`
	`Microsoft.AspNetCore.Builder`
	`Microsoft.AspNetCore.Hosting`
	`Microsoft.AspNetCore.Http`
	`Microsoft.AspNetCore.Routing`
	`Microsoft.Extensions.Configuration`
	`Microsoft.Extensions.DependencyInjection`
	`Microsoft.Extensions.Hosting`
	`Microsoft.Extensions.Logging`

`Microsoft.NET.Sdk.Worker`	Same as `Microsoft.NET.Sdk` and: `Microsoft.Extensions.Configuration`

```
Microsoft.Extensions.DependencyInjection

Microsoft.Extensions.Hosting

Microsoft.Extensions.Logging
```

Let's see the current auto-generated implicit imports file:

- If you are using Visual Studio 2022, then in **Solution Explorer**, select the `Vocabulary` project, toggle on the **Show All Files** button, and note the compiler-generated `bin` and `obj` folders are now visible.

1. Expand the `obj` folder, expand the `Debug` folder, expand the `net7.0` folder, and open the file named `Vocabulary.GlobalUsings.g.cs`.

   The naming convention for this file is `<ProjectName>.GlobalUsings.g.cs`. Note the **g** for **generated** to differentiate from developer-written code files.

2. Remember that this file is automatically created by the compiler for projects that target .NET 6.0, and that it imports some commonly used namespaces including `System.Threading`, as shown in the following code:

```
// <autogenerated />
global using global::System;
global using global::System.Collections.Generic;
global using global::System.IO;
global using global::System.Linq;
global using global::System.Net.Http;
global using global::System.Threading;
global using global::System.Threading.Tasks;
```

1. Close the `Vocabulary.GlobalUsings.g.cs` file.
2. In **Solution Explorer**, select the project, and then add additional entries to the project file to control which namespaces are implicitly imported, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
  <ItemGroup>
    <Using Remove="System.Threading" />
    <Using Include="System.Numerics" />
  </ItemGroup>
</Project>
```

1. Save the changes to the project file.
2. Expand the `obj` folder, expand the `Debug` folder, expand the `net7.0` folder, and open the file named `Vocabulary.GlobalUsings.g.cs`.
3. Note this file now imports `System.Numerics` instead of `System.Threading`, as shown highlighted in the following code:

```
// <autogenerated />
global using global::System;
global using global::System.Collections.Generic;
global using global::System.IO;
global using global::System.Linq;
global using global::System.Net.Http;
global using global::System.Numerics;
global using global::System.Threading.Tasks;
```

1. Close the `Vocabulary.GlobalUsings.g.cs` file.

You can disable the implicitly imported namespaces feature for all SDKs by removing the `<ImplicitUsings>` element completely from the project file, or changing its value to `disable`, as shown in the following markup:

```
<ImplicitUsings>disable</ImplicitUsings>
```

If you are using Visual Studio 2022, then you can control project settings in the user interface.

1. In **Solution Explorer**, right-click the `Vocabulary` project and select **Properties**.
2. Click **Build** and note it has a **General** section opened by default.

3. Scroll down and note the sections to control nullability and implicit imports, as shown in *Figure 2.3*:

Figure 2.3: Controlling project settings in Visual Studio 2022 user interface

4. Close the project properties.

Verbs are methods

In English, verbs are doing or action words, like "run" and "jump". In C#, doing or action words are called **methods**. There are hundreds of thousands of methods available to C#. In English, verbs change how they are written based on when in time the action happens. For example, Amir *was jumping* in the past, Beth *jumps* in the present, they *jumped* in the past, and Charlie *will jump* in the future.

In C#, methods such as `WriteLine` change how they are called or executed based on the specifics of the action. This is called overloading, which we'll cover in more detail in *Chapter 5*, *Building Your Own Types with Object-*

Oriented Programming. But for now, consider the following example:

```
// Outputs the current line terminator string.
// By default, this is a carriage-return and line feed.
Console.WriteLine();
// Outputs the greeting and the current line terminator string.
Console.WriteLine("Hello Ahmed");
// Outputs a formatted number and date and the current line terminator string.
Console.WriteLine("Temperature on {0:D} is {1}˚C.",
  DateTime.Today, 23.4);
```

When I show code snippets without numbered step-by-step instructions for you to enter the code, I do not expect you to enter the code, so it might not execute out-of-context.

A different analogy is that some words are spelled the same but have different meanings depending on the context.

Nouns are types, variables, fields, and properties

In English, nouns are names that refer to things. For example, Fido is the name of a dog. The word "dog" tells us the type of thing that Fido is, and so to order Fido to fetch a ball, we would use his name.

In C#, their equivalents are **types**, **variables**, **fields**, and **properties**. For example:

- `Animal` and `Car` are types; they are nouns for categorizing things.
- `Head` and `Engine` might be fields or properties; nouns that belong to `Animal` and `Car`.
- `Fido` and `Bob` are variables; nouns for referring to a specific object.

There are tens of thousands of types available to C#, though have you noticed how I didn't say, "There are tens of thousands of types *in* C#?" The difference is subtle but important. The language of C# only has a few keywords for types, such as `string` and `int`, and strictly speaking, C# doesn't define any types. Keywords such as `string` that look like types are **aliases**, which represent types provided by the platform on which C# runs.

It's important to know that C# cannot exist alone; after all, it's a language that runs on variants of .NET. In theory, someone could write a compiler for C# that uses a different platform, with different underlying types. In practice, the platform for C# is .NET, which provides tens of thousands of types to C#, including `System.Int32`, which is the C# keyword alias `int` maps to, as well as many more complex types, such as `System.Xml.Linq.XDocument`.

It's worth taking note that the term **type** is often confused with **class**. Have you ever played the parlor game *Twenty Questions*, also known as *Animal, Vegetable, or Mineral*? In the game, everything can be categorized as an animal, vegetable, or mineral. In C#, every **type** can be categorized as a `class`, `struct`, `enum`, `interface`, or `delegate`. You will learn what these mean in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*. As examples, the C# keyword `string` is a `class`, but `int` is a `struct`. So, it is best to use the term **type** to refer to both.

Revealing the extent of the C# vocabulary

We know that there are more than 100 keywords in C#, but how many types are there? Let's write some code to find out how many types (and their methods) are available to C# in our simple console app.

Don't worry about exactly how this code works for now, but know that it uses a technique called **reflection**:

1. Delete all the existing statements in `Program.cs`.
2. We'll start by importing the `System.Reflection` namespace at the top of the `Program.cs` file, as shown in the following code:

```
using System.Reflection;
```

Good Practice: We could use the implicit imports and `global using` features to import this namespace for all `.cs` files in this project, but since there is only one file, it is better to import the namespace in the one file

in which it is needed.

1. Write statements to get the compiled console app and loop through all of the types that it has access to, outputting the names and number of methods each has, as shown in the following code:

```
Assembly? myApp = Assembly.GetEntryAssembly();
if (myApp == null) return; // quit the app
// loop through the assemblies that my app references
foreach (AssemblyName name in myApp.GetReferencedAssemblies())
{
  // load the assembly so we can read its details
  Assembly a = Assembly.Load(name);
  // declare a variable to count the number of methods
  int methodCount = 0;
  // loop through all the types in the assembly
  foreach (TypeInfo t in a.DefinedTypes)
  {
    // add up the counts of methods
    methodCount += t.GetMethods().Count();
  }
  // output the count of types and their methods
  Console.WriteLine(
    "{0:N0} types with {1:N0} methods in {2} assembly.",
    arg0: a.DefinedTypes.Count(),
    arg1: methodCount,
    arg2: name.Name);
}
```

`N0` is upper-case `N` followed by the digit zero. It is not upper-case `N` followed by upper case `O`. It means format a number (`N`) with zero (`0`) decimal places.

1. Run the code. You will see the actual number of types and methods that are available to you in the simplest application when running on your OS. The number of types and methods displayed will be different depending on the operating system that you are using, as shown in the following outputs:

```
// Output on Windows
0 types with 0 methods in System.Runtime assembly.
44 types with 645 methods in System.Console assembly.
106 types with 1,126 methods in System.Linq assembly.
// Output on macOS
0 types with 0 methods in System.Runtime assembly.
57 types with 701 methods in System.Console assembly.
103 types with 1,094 methods in System.Linq assembly.
```

Why does the `System.Runtime` assembly contain zero types? This assembly is special because it contains only **type-forwarders** rather than actual types. A type-forwarder represents a type that has been implemented outside of .NET or for some other advanced reason.

1. Add statements to the top of the file (after importing the namespace) to declare some variables, as shown in the following code:

```
using System.Reflection;
// declare some unused variables using types
// in additional assemblies
System.Data.DataSet ds;
HttpClient client;
```

By declaring variables that use types in other assemblies, those assemblies are loaded with our application, which allows our code to see all the types and methods in them. The compiler will warn you that you have unused variables, but that won't stop your code from running.

1. Run the console app again and view the results, which should look like the following outputs:

```
// Output on Windows
0 types with 0 methods in System.Runtime assembly.
383 types with 6,854 methods in System.Data.Common assembly.
```

```
456 types with 4,590 methods in System.Net.Http assembly.
44 types with 645 methods in System.Console assembly.
106 types with 1,126 methods in System.Linq assembly.
// Output on macOS
0 types with 0 methods in System.Runtime assembly.
376 types with 6,763 methods in System.Data.Common assembly.
522 types with 5,141 methods in System.Net.Http assembly.
57 types with 701 methods in System.Console assembly.
103 types with 1,094 methods in System.Linq assembly.
```

Now, you have a better sense of why learning C# is a challenge, because there are so many types and methods to learn. Methods are only one category of a member that a type can have, and you and other programmers are constantly defining new types and members!

Working with variables

All applications process data. Data comes in, data is processed, and then data goes out.

Data usually comes into our program from files, databases, or user input, and it can be put temporarily into variables that will be stored in the memory of the running program. When the program ends, the data in memory is lost. Data is usually output to files and databases, or to the screen or a printer. When using variables, you should think about, firstly, how much space the variable takes in the memory, and, secondly, how fast it can be processed.

We control this by picking an appropriate type. You can think of simple common types such as `int` and `double` as being different-sized storage boxes, where a smaller box would take less memory but may not be as fast at being processed; for example, adding 16-bit numbers might not be processed as fast as adding 64-bit numbers on a 64-bit operating system. Some of these boxes may be stacked close by, and some may be thrown into a big heap further away.

Naming things and assigning values

There are naming conventions for things, and it is good practice to follow them, as shown in the following table:

Naming convention	Examples	Used for
Camel case	`cost` , `orderDetail` , `dateOfBirth`	Local variables, private fields.
Title case aka Pascal case	`String` , `Int32` , `Cost` , `DateOfBirth` , `Run`	Types, non-private fields, and other members like methods.

Some C# programmers like to prefix the names of private fields with an underscore, for example, `_dateOfBirth` instead of `dateOfBirth`. The naming of private members of all kinds is not formally defined because they will not be visible outside the class, so both are valid. My preference is without an underscore.

Good Practice: Following a consistent set of naming conventions will enable your code to be easily understood by other developers (and yourself in the future!).

The following code block shows an example of declaring a named local variable and assigning a value to it with the `=` symbol. You should note that you can output the name of a variable using a keyword introduced in C# 6.0, `nameof`:

```
// let the heightInMetres variable become equal to the value 1.88
double heightInMetres = 1.88;
Console.WriteLine($"The variable {nameof(heightInMetres)} has the value
{heightInMetres}.");
```

Warning! The message in double quotes in the preceding code wraps onto a second line because the width of a printed page is too narrow. When entering a statement like this in your code editor, type it all in a single line.

Literal values

When you assign to a variable, you often, but not always, assign a **literal** value. But what is a literal value? A literal is a notation that represents a fixed value. Data types have different notations for their literal values, and over the next few sections, you will see examples of using literal notation to assign values to variables.

Storing text

For text, a single letter, such as an `A`, is stored as a `char` type.

> **Good Practice**: Actually, it can be more complicated than that. Egyptian Hieroglyph A002 (U+13001) needs two `System.Char` values (known as surrogate pairs) to represent it: `\uD80C` and `\uDC01`. Do not always assume one `char` equals one letter or you could introduce weird bugs into your code.

A `char` is assigned using single quotes around the literal value, or assigning the return value of a function call, as shown in the following code:

```
char letter = 'A'; // assigning literal characters
char digit = '1';
char symbol = '$';
char userChoice = GetSomeKeystroke(); // assigning from a fictitious function
```

For text, multiple letters, such as `Bob`, are stored as a `string` type and are assigned using double quotes around the literal value, or assigning the return value of a function call, as shown in the following code:

```
string firstName = "Bob"; // assigning literal strings
string lastName = "Smith";
string phoneNumber = "(215) 555-4256";
// assigning a string returned from a fictitious function
string address = GetAddressFromDatabase(id: 563);
// assigning an emoji by converting from Unicode
string grinningEmoji = char.ConvertFromUtf32(0x1F600);
```

To output emoji at the command-line on Windows, you must use Windows Terminal because Command Prompt does not support emoji, and set the output encoding to use UTF-8, as shown in the following code:

```
Console.OutputEncoding = System.Text.Encoding.UTF8;
string grinningEmoji = char.ConvertFromUtf32(0x1F600);
Console.WriteLine(grinningEmoji);
```

Understanding verbatim strings

When storing text in a `string` variable, you can include escape sequences, which represent special characters like tabs and new lines using a backslash, as shown in the following code:

```
string fullNameWithTabSeparator = "Bob\tSmith";
```

But what if you are storing the path to a file on Windows, and one of the folder names starts with a `T`, as shown in the following code?

```
string filePath = "C:\televisions\sony\bravia.txt";
```

The compiler will convert the `\t` into a tab character and you will get errors!

You must prefix with the `@` symbol to use a verbatim literal `string`, as shown in the following code:

```
string filePath = @"C:\televisions\sony\bravia.txt";
```

Understanding raw string literals

Early Access Program edition: To use C# 11 features during .NET 7 previews, you must add an entry to your project file: `<LangVersion>preview</LangVersion>`

Introduced in C# 11, raw string literals are convenient for entering any arbitrary text without needing to escape the contents. They make it easy to define literals containing other languages like XML, HTML, or JSON.

Raw string literals start and end with three or more double-quote characters, as shown in the following code:

```
string xml = """
             <person age="50">
               <first_name>Mark</first_name>
             </person>
             """;
```

In the previous code, the XML is indented by 13 spaces. The compiler looks at the indentation of the last three double-quote characters, and then automatically removes that level of indentation from all the content inside the raw string literal, as shown in the following markup:

```
<person age="50">
  <first_name>Mark</first_name>
</person>
```

You can mix interpolated with raw string literals. You specify the number of braces that indicate a replaced expression by adding that number of dollar signs to the start of the literal. Any fewer braces than that is treated as raw content.

For example, if we want to define some JSON, single braces will be treated as normal braces, but the two dollar symbols tell the compiler that any two curly braces indicate a replaced expression value, as shown in the following code:

```
string json = $$"""
              {
                "first_name": "{{person.FirstName}}",
                "age": {{person.Age}},
              };
              """
```

Summarizing options for storing text

To summarize:

- **Literal string**: Characters enclosed in double-quote characters. They can use escape characters like `\t` for tab. To represent a backslash, use two: `\\`.
- **Raw string literal: Characters enclosed in three or more double-quote characters.**
- **Verbatim string**: A literal string prefixed with `@` to disable escape characters so that a backslash is a backslash. It also allows the `string` value to span multiple lines because the white space characters are treated as themselves instead of instructions to the compiler.
- **Interpolated string**: A literal string prefixed with `$` to enable embedded formatted variables. You will learn more about this later in this chapter.

Storing numbers

Numbers are data that we want to perform an arithmetic calculation on, for example, multiplying. A telephone number is not a number. To decide whether a variable should be stored as a number or not, ask yourself whether you need to perform arithmetic operations on the number or whether the number includes non-digit characters such as parentheses or hyphens to format the number, such as (414) 555-1234. In this case, the "number" is a sequence of characters, so it should be stored as a `string`.

Numbers can be natural numbers, such as 42, used for counting (also called whole numbers); they can also be negative numbers, such as -42 (called integers); or, they can be real numbers, such as 3.9 (with a fractional part),

which are called single- or double-precision floating-point numbers in computing.

Let's explore numbers:

1. Use your preferred code editor to add a new **Console App** / `console` project named `Numbers` to the `Chapter02` workspace/solution.
   - If you are using Visual Studio Code, then select `Numbers` as the active OmniSharp project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.
   - If you are using Visual Studio 2022, then set the startup project to the current selection.
2. In `Program.cs`, delete the existing code and then type statements to declare some number variables using various data types, as shown in the following code:

```
// unsigned integer means positive whole number or 0
uint naturalNumber = 23;
// integer means negative or positive whole number or 0
int integerNumber = -23;
// float means single-precision floating point
// F suffix makes it a float literal
float realNumber = 2.3F;
// double means double-precision floating point
// double is the default type for a number value with a decimal point .
double anotherRealNumber = 2.3; // double literal
```

Storing whole numbers

You might know that computers store everything as bits. The value of a bit is either `0` or `1`. This is called a **binary number system**. Humans use a **decimal number system**.

The decimal number system, also known as Base 10, has 10 as its **base**, meaning there are ten digits, from 0 to 9. Although it is the number base most used by human civilizations, other number base systems are popular in science, engineering, and computing. The binary number system, also known as Base 2, has two as its base, meaning there are two digits, 0 and 1.

The following table shows how computers store the decimal number 10. Take note of the bits with the value 1 in the 8 and 2 columns; 8 + 2 = 10:

128	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0

So, `10` in decimal is `00001010` in binary.

Improving legibility by using digit separators

Two of the improvements seen in C# 7.0 and later are the use of the underscore character `_` as a digit separator, and support for binary literals.

You can insert underscores anywhere into the digits of a number literal, including decimal, binary, or hexadecimal notation, to improve legibility.

For example, you could write the value for 1 million in decimal notation, that is, Base 10, as `1_000_000`.

You can even use the 2/3 grouping common in India: `10_00_000`.

Using binary or hexadecimal notation

To use binary notation, that is, Base 2, using only 1s and 0s, start the number literal with `0b`. To use hexadecimal notation, that is, Base 16, using 0 to 9 and A to F, start the number literal with `0x`.

Exploring whole numbers

Let's enter some code to see some examples:

1. In `Program.cs`, type statements to declare some number variables using underscore separators, as shown in the following code:

```
// three variables that store the number 2 million
int decimalNotation = 2_000_000;
int binaryNotation = 0b_0001_1110_1000_0100_1000_0000;
int hexadecimalNotation = 0x_001E_8480;
// check the three variables have the same value
// both statements output true
Console.WriteLine($"{decimalNotation == binaryNotation}");
Console.WriteLine(
  $"{decimalNotation == hexadecimalNotation}");
```

1. Run the code and note the result is that all three numbers are the same, as shown in the following output:

```
True
True
```

Computers can always exactly represent integers using the `int` type or one of its sibling types, such as `long` and `short`.

Storing real numbers

Computers cannot always represent real, aka decimal or non-integer, numbers precisely. The `float` and `double` types store real numbers using single- and double-precision floating points.

Most programming languages implement the IEEE Standard for Floating-Point Arithmetic. IEEE 754 is a technical standard for floating-point arithmetic established in 1985 by the **Institute of Electrical and Electronics Engineers** (**IEEE**).

The following table shows a simplification of how a computer represents the number `12.75` in binary notation. Note the bits with the value `1` in the 8, 4, ½, and ¼ columns.

8 + 4 + ½ + ¼ = 12¾ = 12.75.

128	64	32	16	8	4	2	1	.	½	¼	1/8	1/16
0	0	0	0	1	1	0	0	.	1	1	0	0

So, `12.75` in decimal is `00001100.1100` in binary. As you can see, the number `12.75` can be exactly represented using bits. However, some numbers can't, something that we'll be exploring shortly.

Writing code to explore number sizes

C# has an operator named `sizeof()` that returns the number of bytes that a type uses in memory. Some types have members named `MinValue` and `MaxValue`, which return the minimum and maximum values that can be stored in a variable of that type. We are now going to use these features to create a console app to explore number types:

1. In `Program.cs`, type statements to show the size of three number data types, as shown in the following code:

```
Console.WriteLine($"int uses {sizeof(int)} bytes and can store numbers in the range {int.MinValue
Console.WriteLine($"double uses {sizeof(double)} bytes and can store numbers in the range {double
Console.WriteLine($"decimal uses {sizeof(decimal)} bytes and can store numbers in the range {dec
```

Warning! The width of the printed pages in this book makes the `string` values (in double quotes) wrap over multiple lines. You must type them on a single line, or you will get compile errors.

1. Run the code and view the output, as shown in *Figure 2.4*:

Figure 2.4: Size and range information for common number data types

An `int` variable uses four bytes of memory and can store positive or negative numbers up to about 2 billion. A `double` variable uses eight bytes of memory and can store much bigger values! A `decimal` variable uses 16 bytes of memory and can store big numbers, but not as big as a `double` type.

But you may be asking yourself, why might a `double` variable be able to store bigger numbers than a `decimal` variable, yet it's only using half the space in memory? Well, let's now find out!

Comparing double and decimal types

You will now write some code to compare `double` and `decimal` values. Although it isn't hard to follow, don't worry about understanding the syntax right now:

1. Type statements to declare two `double` variables, add them together and compare them to the expected result, and write the result to the console, as shown in the following code:

```
Console.WriteLine("Using doubles:");
double a = 0.1;
double b = 0.2;
if (a + b == 0.3)
{
  Console.WriteLine($"{a} + {b} equals {0.3}");
}
else
{
  Console.WriteLine($"{a} + {b} does NOT equal {0.3}");
}
```

1. Run the code and view the result, as shown in the following output:

```
Using doubles:
0.1 + 0.2 does NOT equal 0.3
```

In locales that use a comma for the decimal separator the result will look slightly different, as shown in the following output:

```
0,1 + 0,2 does NOT equal 0,3
```

The `double` type is not guaranteed to be accurate because some numbers like `0.1` literally cannot be represented as floating-point values.

As a rule of thumb, you should only use `double` when accuracy, especially when comparing the equality of two numbers, is not important. An example of this may be when you're measuring a person's height and you will only compare values using greater than or less than, but never equals.

The problem with the preceding code is illustrated by how the computer stores the number `0.1`, or multiples of it. To represent `0.1` in binary, the computer stores 1 in the 1/16 column, 1 in the 1/32 column, 1 in the 1/256 column, 1 in the 1/512 column, and so on.

The number `0.1` in decimal is `0.00011001100110011`… in binary, repeating forever:

4	2	1	.	½	¼	1/8	1/16	1/32	1/64	1/128	1/256	1/512	1/1024	1/2048
0	0	0	.	0	0	0	1	1	0	0	1	1	0	0

> **Good Practice**: Never compare `double` values using `==`. During the First Gulf War, an American Patriot missile battery used `double` values in its calculations. The inaccuracy caused it to fail to track and intercept an incoming Iraqi Scud missile, and 28 soldiers were killed; you can read about this at https://www.ima.umn.edu/~arnold/disasters/patriot.xhtml.

1. Copy and paste the statements that you wrote before (that used the `double` variables).
2. Modify the statements to use `decimal` and rename the variables to `c` and `d`, as shown in the following code:

```
Console.WriteLine("Using decimals:");
decimal c = 0.1M; // M suffix means a decimal literal value
decimal d = 0.2M;
if (c + d == 0.3M)
{
  Console.WriteLine($"{c} + {d} equals {0.3M}");
}
else
{
  Console.WriteLine($"{c} + {d} does NOT equal {0.3M}");
}
```

1. Run the code and view the result, as shown in the following output:

```
Using decimals:
0.1 + 0.2 equals 0.3
```

The `decimal` type is accurate because it stores the number as a large integer and shifts the decimal point. For example, `0.1` is stored as `1`, with a note to shift the decimal point one place to the left. `12.75` is stored as `1275`, with a note to shift the decimal point two places to the left.

> **Good Practice**: Use `int` for whole numbers. Use `double` for real numbers that will not be compared for equality to other values; it is okay to compare `double` values being less than or greater than, and so on. Use `decimal` for money, CAD drawings, general engineering, and wherever the accuracy of a real number is important.

The `float` and `double` type have some useful special values: `NaN` represents not-a-number (for example, the result of dividing by zero), `Epsilon` represents the smallest positive number that can be stored in a `float` or `double`, and `PositiveInfinity` and `NegativeInfinity` represent infinitely large positive and negative values. They also have methods for checking for these special values like `IsInfinity` and `IsNan`.

Storing Booleans

Booleans can only contain one of the two literal values `true` or `false`, as shown in the following code:

```
bool happy = true;
bool sad = false;
```

They are most used to branch and loop. You don't need to fully understand them yet, as they are covered more in *Chapter 3*, *Controlling Flow, Converting Types, and Handling Exceptions*.

Storing any type of object

There is a special type named `object` that can store any type of data, but its flexibility comes at the cost of messier code and possibly poor performance. Because of those two reasons, you should avoid it whenever possible. The following steps show how to use object types if you need to use them:

1. Use your preferred code editor to add a new **Console App** / `console` project named `Variables` to the `Chapter02` workspace/solution.
   - If you are using Visual Studio Code, then select `Variables` as the active OmniSharp project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.
2. In `Program.cs`, delete the existing statements and then type statements to declare and use some variables using the `object` type, as shown in the following code:

```
object height = 1.88; // storing a double in an object
object name = "Amir"; // storing a string in an object
Console.WriteLine($"{name} is {height} metres tall.");
int length1 = name.Length; // gives compile error!
int length2 = ((string)name).Length; // tell compiler it is a string
Console.WriteLine($"{name} has {length2} characters.");
```

1. Run the code and note that the fourth statement cannot compile because the data type of the `name` variable is not known by the compiler, as shown in *Figure 2.5*:

Figure 2.5: The object type does not have a Length property

2. Add comment double slashes to the beginning of the statement that cannot compile to comment out the statement, making it inactive.
3. Run the code again and note that the compiler can access the length of a `string` if the programmer explicitly tells the compiler that the `object` variable contains a `string` by prefixing with a cast expression like `(string)`, as shown in the following output:

```
Amir is 1.88 metres tall.
Amir has 4 characters.
```

The `object` type has been available since the first version of C#, but C# 2.0 and later have a better alternative called **generics**, which we will cover in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*, which will provide us with the flexibility we want, but without the performance overhead.

Storing dynamic types

There is another special type named `dynamic` that can also store any type of data, but even more than `object`, its flexibility comes at the cost of performance. The `dynamic` keyword was introduced in C# 4.0. However, unlike `object`, the value stored in the variable can have its members invoked without an explicit cast. Let's make use of a `dynamic` type:

1. Add statements to declare a `dynamic` variable and then assign a `string` literal value, and then an integer value, and then an array of integer values, as shown in the following code:

```
// storing a string in a dynamic object
// string has a Length property
dynamic something = "Ahmed";
// int does not have a Length property
// something = 12;
// an array of any type has a Length property
// something = new[] { 3, 5, 7 };
```

1. Add a statement to output the length of the `dynamic` variable, as shown in the following code:

```
// this compiles but would throw an exception at run-time
// if you later stored a data type that does not have a
// property named Length
Console.WriteLine($"Length is {something.Length}");
```

1. Run the code and note it works because a `string` value does have a `Length` property, as shown in the following output:

```
Length is 5
```

1. Uncomment the statement that assigns an `int` value of 12 to the `something` variable.
2. Run the code and note the runtime error because `int` does not have a `Length` property, as shown in the following output:

```
Unhandled exception. Microsoft.CSharp.RuntimeBinder.RuntimeBinderException: 'int' does not conta
```

1. Uncomment the statement that assigns the array of three integers 3, 5, and 7 to the `something` variable.
2. Run the code and note the output because an array of three `int` values does have a `Length` property, as shown in the following output:

```
Length is 3
```

One limitation of `dynamic` is that code editors cannot show IntelliSense to help you write the code. This is because the compiler cannot check what the type is during build time. Instead, the CLR checks for the member at runtime and throws an exception if it is missing.

Exceptions are a way to indicate that something has gone wrong at runtime. You will learn more about them and how to handle them in *Chapter 3*, *Controlling Flow, Converting Types, and Handling Exceptions*.

Declaring local variables

Local variables are declared inside methods, and they only exist during the execution of that method. Once the method returns, the memory allocated to any local variables is released.

Strictly speaking, value types are released while reference types must wait for a garbage collection. You will learn about the difference between value types and reference types in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*.

Specifying the type of a local variable

Let's explore local variables declared with specific types and using type inference:

Type statements to declare and assign values to some local variables using specific types, as shown in the following code:

```
int population = 67_000_000; // 67 million in UK
double weight = 1.88; // in kilograms
decimal price = 4.99M; // in pounds sterling
string fruit = "Apples"; // strings use double-quotes
char letter = 'Z'; // chars use single-quotes
bool happy = true; // Booleans have value of true or false
```

Depending on your code editor and color scheme, it will show green squiggles under each of the variable names and lighten their text color to warn you that the variable is assigned but its value is never used.

Inferring the type of a local variable

You can use the `var` keyword to declare local variables with C# 3 and later. The compiler will infer the type from the value that you assign after the assignment operator, `=`.

A literal number without a decimal point is inferred as an `int` variable, that is, unless you add a suffix, as described in the following list:

- `L`: infers `long`
- `UL`: infers `ulong`
- `M`: infers `decimal`
- `D`: infers `double`
- `F`: infers `float`

A literal number with a decimal point is inferred as `double` unless you add the `M` suffix, in which case, it infers a `decimal` variable, or the `F` suffix, in which case, it infers a `float` variable.

Double quotes indicate a `string` variable, single quotes indicate a `char` variable, and the `true` and `false` values infer a `bool` type:

1. Modify the previous statements to use `var`, as shown in the following code:

```
var population = 67_000_000; // 67 million in UK
var weight = 1.88; // in kilograms
var price = 4.99M; // in pounds sterling
var fruit = "Apples"; // strings use double-quotes
var letter = 'Z'; // chars use single-quotes
var happy = true; // Booleans have value of true or false
```

1. Hover your mouse over each of the `var` keywords and note that your code editor shows a tooltip with information about the type that has been inferred.
2. At the top of `Program.cs`, import the namespace for working with XML to enable us to declare some variables using types in that namespace, as shown in the following code:

```
using System.Xml;
```

Good Practice: If you are using .NET Interactive Notebooks, then add `using` statements in a separate code cell above the code cell where you write the main code. Then click **Execute Cell** to ensure the namespaces are imported. They will then be available in subsequent code cells.

1. Under the previous statements, add statements to create some new objects, as shown in the following code:

```
// good use of var because it avoids the repeated type
// as shown in the more verbose second statement
var xml1 = new XmlDocument(); // C# 3 and later
XmlDocument xml2 = new XmlDocument(); // all C# versions
// bad use of var because we cannot tell the type, so we
// should use a specific type declaration as shown in
// the second statement
```

```
var file1 = File.CreateText("something1.txt");
StreamWriter file2 = File.CreateText("something2.txt");
```

> **Good Practice**: Although using `var` is convenient, some developers avoid using it, to make it easier for a code reader to understand the types in use. Personally, I use it only when the type is obvious. For example, in the preceding code statements, the first statement is just as clear as the second in stating what the type of the `xml` variables are, but it is shorter. However, the third statement isn't clear in showing the type of the `file` variable, so the fourth is better because it shows that the type is `StreamWriter`. If in doubt, spell it out!

Using target-typed new to instantiate objects

With C# 9, Microsoft introduced another syntax for instantiating objects known as **target-typed new**. When instantiating an object, you can specify the type first and then use `new` without repeating the type, as shown in the following code:

```
XmlDocument xml3 = new(); // target-typed new in C# 9 or later
```

If you have a type with a field or property that needs to be set, then the type can be inferred, as shown in the following code:

```
// In Program.cs
Person kim = new();
kim.BirthDate = new(1967, 12, 26); // instead of: new DateTime(1967, 12, 26)
// In a separate Person.cs file or at the bottom of Program.cs
class Person
{
  public DateTime BirthDate;
}
```

This way of instantiating objects is especially useful with arrays and collections because they have multiple objects often of the same type, as shown in the following code:

```
List<Person> people = new()
{
  new() { FirstName = "Alice" },
  new() { FirstName = "Bob" },
  new() { FirstName = "Charlie" }
};
```

You will learn about arrays in *Chapter 3*, *Controlling Flow, Converting Types, and Handling Exceptions*, and collections in *Chapter 8*, *Working with Common .NET Types*.

> **Good Practice**: Use target-typed new to instantiate objects unless you must use a pre-version 9 C# compiler. I have used target-typed new throughout the rest of this book. Please let me know if you spot any cases that I missed!

Getting and setting the default values for types

Most of the primitive types except `string` are **value types**, which means that they must have a value. You can determine the default value of a type by using the `default()` operator and passing the type as a parameter. You can assign the default value of a type by using the `default` keyword.

The `string` type is a **reference type**. This means that `string` variables contain the memory address of a value, not the value itself. A reference type variable can have a `null` value, which is a literal that indicates that the variable does not reference anything (yet). `null` is the default for all reference types.

You'll learn more about value types and reference types in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*.

Let's explore default values:

1. Add statements to show the default values of an `int`, `bool`, `DateTime`, and `string`, as shown in the following code:

```
Console.WriteLine($"default(int) = {default(int)}");
Console.WriteLine($"default(bool) = {default(bool)}");
Console.WriteLine($"default(DateTime) = {default(DateTime)}");
Console.WriteLine($"default(string) = {default(string)}");
```

1. Run the code and view the result, noting that your output for the date and time might be formatted differently if you are not running it in the UK, and that `null` values output as an empty `string`, as shown in the following output:

```
default(int) = 0
default(bool) = False
default(DateTime) = 01/01/0001 00:00:00
default(string) =
```

1. Add statements to declare a number, assign a value, and then reset it to its default value, as shown in the following code:

```
int number = 13;
Console.WriteLine($"number has been set to: {number}");
number = default;
Console.WriteLine($"number has been reset to its default: {number}");
```

1. Run the code and view the result, as shown in the following output:

```
number has been set to: 13
number has been reset to its default: 0
```

Exploring more about console apps

We have already created and used basic console apps, but we're now at a stage where we should delve into them more deeply.

Console apps are text-based and are run at the command line. They typically perform simple tasks that need to be scripted, such as compiling a file or encrypting a section of a configuration file.

Equally, they can also have arguments passed to them to control their behavior.

An example of this would be to create a new console app using the F# language with a specified name instead of using the name of the current folder, as shown in the following command line:

```
dotnet new console -lang "F#" --name "ExploringConsole"
```

Displaying output to the user

The two most common tasks that a console app performs are writing and reading data. We have already been using the `WriteLine` method to output, but if we didn't want a carriage return at the end of a line, for example, if you later want to continue to write more text to the end of that line, we could have used the `Write` method.

Formatting using numbered positional arguments

One way of generating formatted strings is to use numbered positional arguments.

This feature is supported by methods like `Write` and `WriteLine`, and for methods that do not support the feature, the `string` parameter can be formatted using the `Format` method of `string`.

Let's begin formatting:

1. Use your preferred code editor to add a new **Console App** / `console` project named `Formatting` to the `Chapter02` workspace/solution.
   - If you are using Visual Studio Code, then select `Formatting` as the active OmniSharp project.
2. In `Program.cs`, delete the existing statements and then type statements to declare some number variables and write them to the console, as shown in the following code:

```
int numberOfApples = 12;
decimal pricePerApple = 0.35M;
Console.WriteLine(
  format: "{0} apples cost {1:C}",
  arg0: numberOfApples,
  arg1: pricePerApple * numberOfApples);
string formatted = string.Format(
  format: "{0} apples cost {1:C}",
  arg0: numberOfApples,
  arg1: pricePerApple * numberOfApples);
//WriteToFile(formatted); // writes the string into a file
```

The `WriteToFile` method is a nonexistent method used to illustrate the idea.

Good Practice: Once you become more comfortable with formatting strings, you should stop naming the parameters, for example, stop using `format:`, `arg0:`, and `arg1:`. The preceding code uses a non-canonical style to show where the `0` and `1` came from while you are learning.

Formatting using interpolated strings

C# 6.0 and later have a handy feature named **interpolated strings**. A `string` prefixed with `$` can use curly braces around the name of a variable or expression to output the current value of that variable or expression at that position in the `string`, as the following shows:

1. Enter a statement at the bottom of the `Program.cs` file, as shown in the following code:

```
Console.WriteLine($"{numberOfApples} apples cost {pricePerApple * numberOfApples:C}");
```

1. Run the code and view the result, as shown in the following partial output:

```
12 apples cost £4.20
```

For short formatted `string` values, an interpolated `string` can be easier for people to read. But for code examples in a book, where lines need to wrap over multiple lines, this can be tricky. For many of the code examples in this book, I will use numbered positional arguments.

Another reason to avoid interpolated strings is that they can't be read from resource files to be localized.

Before C# 10, `string` constants could only be combined by using concatenation with the `+` operator, as shown in the following code:

```
private const string firstname = "Omar";
private const string lastname = "Rudberg";
private const string fullname = firstname + " " + lastname;
```

With C# 10, interpolated strings (prefixed with `$`) can now be used, as shown in the following code:

```
private const string fullname = $"{firstname} {lastname}";
```

This only works for combining `string` constant values. It cannot work with other types like numbers that would require runtime data type conversions.

Understanding format strings

A variable or expression can be formatted using a format string after a comma or colon.

An `N0` format string means a number with thousand separators and no decimal places, while a `C` format string means currency. The currency format will be determined by the current thread.

For instance, if you run code that uses the number or currency format on a PC in the UK, you'll get pounds sterling with commas as the thousand separators, but if you run it on a PC in Germany, you will get euros with dots as the thousand separators.

The full syntax of a format item is:

```
{ index [, alignment ] [ : formatString ] }
```

Each format item can have an alignment, which is useful when outputting tables of values, some of which might need to be left- or right-aligned within a width of characters. Alignment values are integers. Positive integers mean right-aligned and negative integers mean left-aligned.

For example, to output a table of fruit and how many of each there are, we might want to left-align the names within a column of 10 characters and right-align the counts formatted as numbers with zero decimal places within a column of six characters:

1. At the bottom of `Program.cs`, enter the following statements:

```
string applesText = "Apples";
int applesCount = 1234;
string bananasText = "Bananas";
int bananasCount = 56789;
Console.WriteLine(
  format: "{0,-10} {1,6}",
  arg0: "Name",
  arg1: "Count");
Console.WriteLine(
  format: "{0,-10} {1,6:N0}",
  arg0: applesText,
  arg1: applesCount);
Console.WriteLine(
  format: "{0,-10} {1,6:N0}",
  arg0: bananasText,
  arg1: bananasCount);
```

1. Run the code and note the effect of the alignment and number format, as shown in the following output:

```
Name          Count
Apples        1,234
Bananas      56,789
```

Getting text input from the user

We can get text input from the user using the `ReadLine` method. This method waits for the user to type some text, then as soon as the user presses Enter, whatever the user has typed is returned as a `string` value.

> **Good Practice**: If you are using a .NET Interactive notebook for this section, then note that it does not support reading input from the console using `Console.ReadLine()`. Instead, you must set literal values, as shown in the following code: `string? firstName = "Gary";`. This is often quicker to experiment with because you can simply change the literal `string` value and click the **Execute Cell** button instead of having to restart a console app each time you want to enter a different `string` value.

Let's get input from the user:

1. Type statements to ask the user for their name and age and then output what they entered, as shown in the following code:

```
Console.Write("Type your first name and press ENTER: ");
string firstName = Console.ReadLine();
```

```
Console.Write("Type your age and press ENTER: ");
string age = Console.ReadLine();
Console.WriteLine(
  $"Hello {firstName}, you look good for {age}.");
```

By default, with .NET 6 and later, nullability checks are enabled so the C# compiler gives two warnings because the `ReadLine` method could return a `null` value instead of a `string` value.

1. For the `firstName` variable, append a `?` after `string`. This tells the compiler that you are expecting a possible `null` value so it does not need to warn us. If the variable is `null` then when it is later output with `WriteLine`, it will just be blank, so that works fine in this case. If we were going to access any of the members of the `firstName` variable, then we would need to handle the case where it is `null`.

2. For the `firstName` variable, append a `!` before the semi-colon at the end of the statement. This is called the **null-forgiving operator** because it tells the compiler that in this case `ReadLine` will not return `null` so it can stop showing the warning. It is now our responsibility to ensure this is the case. Luckily, the `Console` type's implementation of `ReadLine` always returns a `string` even if it is just an empty `string` value.

   You have now seen two common ways to handle nullability warnings from the compiler. We will cover nullability and how to handle it in more detail in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*.

3. Run the code, and then enter a name and age, as shown in the following output:

```
Type your name and press ENTER: Gary
Type your age and press ENTER: 34
Hello Gary, you look good for 34.
```

Simplifying the usage of the console

In C# 6.0 and later, the `using` statement can be used not only to import a namespace but also to further simplify our code by importing a static class. Then, we won't need to enter the `Console` type name throughout our code. You can use your code editor's Find and Replace feature to remove the times we have previously written `Console`:

1. At the top of the `Program.cs` file, add a statement to **statically import** the `System.Console` class, as shown in the following code:

```
using static System.Console;
```

1. Select the first `Console.` in your code, ensuring that you select the dot after the word `Console` too.

2. In Visual Studio, navigate to **Edit** | **Find and Replace** | **Quick Replace**, or in Visual Studio Code, navigate to **Edit** | **Replace**, and note that an overlay dialog appears ready for you to enter what you would like to replace **Console.** with, as shown in *Figure 2.6*:

Figure 2.6: Using the Replace feature in Visual Studio to simplify your code

3. Leave the replace box empty, click on the **Replace all** button (the second of the two buttons to the right of the replace box), and then close the replace box by clicking on the cross in its top-right corner.
4. Run the console app and note the behavior is the same as before.
5. Delete the statement to statically import `System.Console`.
6. Open `Formatting.csproj`, and after the `<PropertyGroup>` section, add a new `<ItemGroup>` section to statically import `System.Console` using the implicit usings .NET SDK feature, as shown in the following markup:

```
<ItemGroup>
  <Using Include="System.Console" Static="true" />
</ItemGroup>
```

1. Run the console app and note the behavior is the same as before.

Good Practice: In future, for all console app projects you create for this book, add the section above to simplify the code you need to write in all C# files to work with the `Console` class.

Getting key input from the user

We can get key input from the user using the `ReadKey` method. This method waits for the user to press a key or key combination that is then returned as a `ConsoleKeyInfo` value.

You will not be able to execute the call to the `ReadKey` method using a .NET Interactive notebook, but if you have created a console app, then let's explore reading key presses:

1. Type statements to ask the user to press any key combination and then output information about it, as shown in the following code:

```
Write("Press any key combination: ");
ConsoleKeyInfo key = ReadKey();
WriteLine();
WriteLine("Key: {0}, Char: {1}, Modifiers: {2}",
  arg0: key.Key,
  arg1: key.KeyChar,
  arg2: key.Modifiers);
```

1. Run the code, press the K key, and note the result, as shown in the following output:

```
Press any key combination: k
Key: K, Char: k, Modifiers: 0
```

1. Run the code, hold down Shift and press the K key, and note the result, as shown in the following output:

```
Press any key combination: K
Key: K, Char: K, Modifiers: Shift
```

1. Run the code, press the F12 key, and note the result, as shown in the following output:

```
Press any key combination:
Key: F12, Char: , Modifiers: 0
```

When running a console app in a terminal within Visual Studio Code, some keyboard combinations will be captured by the code editor or operating system before they can be processed by your console app.

Passing arguments to a console app

When you run a console app, you often want to change its behavior by passing arguments. For example, with the `dotnet` command line tool, you can pass the name of a new project template, as shown in the following commands:

```
dotnet new console
dotnet new mvc
```

You might have been wondering how to get any arguments that might be passed to a console app.

In every version of .NET prior to version 6.0, the console app project template made it obvious, as shown in the following code:

```
using System;
namespace Arguments
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Hello World!");
    }
  }
}
```

The `string[] args` arguments are declared and passed in the `Main` method of the `Program` class. They're an array used to pass arguments into a console app. But in top-level programs, as used by the console app project template in .NET 6.0 and later, the `Program` class and its `Main` method are hidden, along with the declaration of the `args` array. The trick is that you must know it still exists.

Command-line arguments are separated by spaces. Other characters like hyphens and colons are treated as part of an argument value.

To include spaces in an argument value, enclose the argument value in single or double quotes.

Imagine that we want to be able to enter the names of some colors for the foreground and background, and the dimensions of the terminal window at the command line. We would be able to read the colors and numbers by reading them from the `args` array, which is always passed into the `Main` method, aka the entry point of a console app:

1. Use your preferred code editor to add a new **Console App** / `console` project named `Arguments` to the `Chapter02` workspace/solution. You will not be able to use a .NET Interactive notebook because you cannot pass arguments to a notebook.
   - If you are using Visual Studio Code, then select `Arguments` as the active OmniSharp project.
2. Open `Arguments.csproj`, and after the `<PropertyGroup>` section, add a new `<ItemGroup>` section to statically import `System.Console` for all C# files using the implicit usings .NET SDK feature, as shown in the following markup:

```
<ItemGroup>
  <Using Include="System.Console" Static="true" />
</ItemGroup>
```

Good Practice: Remember to use the implicit usings .NET SDK feature to statically import the `System.Console` type in all future console app projects to simplify your code, as these instructions will not be repeated every time.

1. In `Program.cs`, delete the existing statements and then add a statement to output the number of arguments passed to the application, as shown in the following code:

```
WriteLine($"There are {args.Length} arguments.");
```

1. Run the console app and view the result, as shown in the following output:

```
There are 0 arguments.
```

If you are using Visual Studio 2022:

1. Navigate to **Project | Arguments Properties**, select the **Debug** tab, click **Open debug launch profiles UI**, and in the **Command line arguments** box, enter the following arguments: `firstarg second-arg third:arg "fourth arg"`, as shown in *Figure 2.7*:

Figure 2.7: Entering command line arguments in Visual Studio project properties

2. Close the **Launch Profiles** window.
3. Run the console app.

If you are using Visual Studio Code:

- In **Terminal**, enter some arguments after the `dotnet run` command, as shown in the following command:

```
dotnet run firstarg second-arg third:arg "fourth arg"
```

For both coding tools:

1. Note the result indicates four arguments, as shown in the following output:

```
There are 4 arguments.
```

1. In `Program.cs`, to enumerate or iterate (that is, loop through) the values of those four arguments, add the following statements after outputting the length of the array:

```
foreach (string arg in args)
{
  WriteLine(arg);
}
```

1. Run the code again and note the result shows the details of the four arguments, as shown in the following output:

```
There are 4 arguments.
firstarg
second-arg
third:arg
fourth arg
```

Setting options with arguments

We will now use these arguments to allow the user to pick a color for the background, foreground, and cursor size of the output window. The cursor size can be an integer value from 1, meaning a line at the bottom of the cursor cell, up to 100, meaning a percentage of the height of the cursor cell.

We have statically imported the `System.Console` class. It has properties like `ForegroundColor`, `BackgroundColor`, and `CursorSize` that we can now set just by using their names without needing to prefix with `Console.`

The `System` namespace is already imported so that the compiler knows about the `ConsoleColor` and `Enum` types:

1. Add statements to warn the user if they do not enter three arguments and then parse those arguments and use them to set the color and dimensions of the console window, as shown in the following code:

```
if (args.Length < 3)
{
  WriteLine("You must specify two colors and cursor size, e.g.");
  WriteLine("dotnet run red yellow 50");
  return; // stop running
}
ForegroundColor = (ConsoleColor)Enum.Parse(
  enumType: typeof(ConsoleColor),
  value: args[0],
  ignoreCase: true);
BackgroundColor = (ConsoleColor)Enum.Parse(
  enumType: typeof(ConsoleColor),
  value: args[1],
  ignoreCase: true);
CursorSize = int.Parse(args[2]);
```

Note the compiler warning that setting the `CursorSize` is only supported on Windows.

- If you are using Visual Studio 2022, then navigate to **Project | Arguments Properties**, and change the arguments to `red yellow 50`, run the console app, and note the cursor is half the size and the colors have changed in the window, as shown in *Figure 2.8*:

Figure 2.8: Setting colors and cursor size on Windows

- If you are using Visual Studio Code, then run the code with arguments to set the foreground color to red, the background color to yellow, and the cursor size to 50%, as shown in the following command:

```
dotnet run red yellow 50
```

On macOS, you'll see an unhandled exception, as shown in *Figure 2.9*:

Figure 2.9: An unhandled exception on unsupported macOS

Although the compiler did not give an error or warning, at runtime some API calls may fail on some platforms. Although a console app running on Windows can change its cursor size, on macOS, it cannot, and complains if you try.

Handling platforms that do not support an API

So how do we solve this problem? We can solve this by using an exception handler. You will learn more details about the `try-catch` statement in *Chapter 3*, *Controlling Flow, Converting Types, and Handling Exceptions*, so for now, just enter the code:

1. Modify the code to wrap the lines that change the cursor size in a `try` statement, as shown in the following code:

```
try
{
  CursorSize = int.Parse(args[2]);
}
catch (PlatformNotSupportedException)
{
  WriteLine("The current platform does not support changing the size of the cursor.");
}
```

1. If you were to run the code on macOS then you would see the exception is caught, and a friendlier message is shown to the user.

Another way to handle differences in operating systems is to use the `OperatingSystem` class in the `System` namespace, as shown in the following code:

```
if (OperatingSystem.IsWindows())
{
  // execute code that only works on Windows
}
else if (OperatingSystem.IsWindowsVersionAtLeast(major: 10))
{
  // execute code that only works on Windows 10 or later
}
else if (OperatingSystem.IsIOSVersionAtLeast(major: 14, minor: 5))
{
  // execute code that only works on iOS 14.5 or later
}
else if (OperatingSystem.IsBrowser())
{
  // execute code that only works in the browser with Blazor
}
```

The `OperatingSystem` class has equivalent methods for other common operating systems like Android, iOS, Linux, macOS, and even the browser, which is useful for Blazor web components.

A third way to handle different platforms is to use conditional compilation statements.

There are four preprocessor directives that control conditional compilation: `#if`, `#elif`, `#else`, and `#endif`.

You define symbols using `#define`, as shown in the following code:

```
#define MYSYMBOL
```

Many symbols are automatically defined for you, as shown in the following table:

Target Framework	**Symbols**
.NET Standard	`NETSTANDARD2_0`, `NETSTANDARD2_1`, and so on
Modern .NET	`NET7_0`, `NET7_0_ANDROID`, `NET7_0_IOS`, `NET7_0_WINDOWS`, and so on

You can then write statements that will compile only for the specified platforms, as shown in the following code:

```
#if NET7_0_ANDROID
// compile statements that only works on Android
#elif NET7_0_IOS
// compile statements that only works on iOS
#else
// compile statements that work everywhere else
#endif
```

Understanding async and await

C# 5 introduced two C# keywords when working with the `Task` type that enables easy multi-threading. The pair of keywords are especially useful for the following:

- Implementing multitasking for a **graphical user interface** (**GUI**).
- Improving the scalability of web applications and web services.
- Preventing blocking calls when interacting with the filesystem, databases, and remote services, all of which tend to take a long time to complete their work.

In *Chapter 13*, *Building Websites Using the Model-View-Controller Pattern*, we will see how the `async` and `await` keywords can improve scalability for websites.

But for now, let's learn the theory of why these two C# keywords were introduced, and then later you will see them used in practice.

Improving responsiveness for console apps

One of the limitations with console apps is that you can only use the `await` keyword inside methods that are marked as `async`, but C# 7 and earlier do not allow the `Main` method to be marked as `async`! Luckily, a new feature introduced in C# 7.1 was support for `async` in `Main`.

Let's see it in action:

1. Use your preferred code editor to add a new **Console App** / console project named `AsyncConsole` to the `Chapter02` solution/workspace.
   - If you are using Visual Studio Code, then select `AsyncConsole` as the active OmniSharp project.
2. Open `AsyncConsole.csproj`, and after the `<PropertyGroup>` section, add a new `<ItemGroup>` section to statically import `System.Console` for all C# files using the implicit usings .NET SDK feature, as shown in the following markup:

```
<ItemGroup>
  <Using Include="System.Console" Static="true" />
</ItemGroup>
```

1. In `Program.cs`, delete the existing statements and then add statements to create an `HttpClient` instance, make a request for Apple's home page, and output how many bytes it has, as shown in the following code:

```
HttpClient client = new();
HttpResponseMessage response =
  await client.GetAsync("http://www.apple.com/");
WriteLine("Apple's home page has {0:N0} bytes.",
  response.Content.Headers.ContentLength);
```

1. Navigate to **Build** | **Build AsyncConsole** and note that the project builds successfully.

   In .NET 5 and earlier, you would have seen an error message, as shown in the following output:

   ```
   Program.cs(14,9): error CS4033: The 'await' operator can only be used within an async me
   ```

   You would have had to add the `async` keyword to your `Main` method and change its return type from `void` to `Task`. With .NET 6 and later, the console app project template uses the top-level program feature to automatically define the `Program` class with an asynchronous `<Main>$` method for you.

2. Run the code and view the result, which is likely to have a different number of bytes since Apple changes its home page frequently, as shown in the following output:

```
Apple's home page has 40,252 bytes.
```

Practicing and exploring

Test your knowledge and understanding by answering some questions, getting some hands-on practice, and exploring the topics covered in this chapter with deeper research.

Exercise 2.1 – Test your knowledge

To get the best answer to some of these questions, you will need to do your own research. I want you to "think outside the book" so I have deliberately not provided all the answers in the book.

I want to encourage you to get into the good habit of looking for help elsewhere, following the principle of "teach a person to fish."

1. What statement can you type in a C# file to discover the compiler and language version?
2. What are the two types of comments in C#?
3. What is the difference between a verbatim string and an interpolated string?
4. Why should you be careful when using `float` and `double` values?
5. How can you determine how many bytes a type like `double` uses in memory?
6. When should you use the `var` keyword?
7. What is the newest syntax to create an instance of a class like `XmlDocument`?
8. Why should you be careful when using the `dynamic` type?
9. How do you right-align a format string?
10. What character separates arguments for a console app?

Appendix, *Answers to the Test Your Knowledge Questions* is available to download from a link in the README on the GitHub repository: https://github.com/markjprice/cs11dotnet7.

Exercise 2.2 – Test your knowledge of number types

What type would you choose for the following "numbers"?

1. A person's telephone number
2. A person's height
3. A person's age
4. A person's salary
5. A book's ISBN
6. A book's price
7. A book's shipping weight
8. A country's population
9. The number of stars in the universe
10. The number of employees in each of the small or medium businesses in the United Kingdom (up to about 50,000 employees per business)

Exercise 2.3 – Practice number sizes and ranges

In the `Chapter02` solution/workspace, create a console app project named `Ch02Ex03Numbers` that outputs the number of bytes in memory that each of the following number types uses and the minimum and maximum values they can have: `sbyte`, `byte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`, `float`, `double`, and `decimal`.

The result of running your console app should look something like *Figure 2.10*:

Figure 2.10: The result of outputting number type sizes

Code solutions for all exercises are available to download or clone from the GitHub repository at the following link: https://github.com/markjprice/cs11dotnet7.

Exercise 2.4 – Explore topics

Use the links on the following page to learn more detail about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-2---speaking-c

Summary

In this chapter, you learned how to:

- Declare variables with a specified or an inferred type.
- Use some of the built-in types for numbers, text, and Booleans.
- Choose between number types.

- Control output formatting in console apps.

In the next chapter, you will learn about operators, branching, looping, converting between types, and how to handle exceptions.

3 Controlling Flow, Converting Types, and Handling Exceptions

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is all about writing code that performs simple operations on variables, makes decisions, performs pattern matching, repeats statements or blocks, works with arrays to store multiple values, converts variable or expression values from one type to another, handles exceptions, and checks for overflows in number variables.

This chapter covers the following topics:

- Operating on variables
- Understanding selection statements
- Understanding iteration statements
- Storing multiple values in arrays
- Casting and converting between types
- Handling exceptions
- Checking for overflow

Operating on variables

Operators apply simple operations such as addition and multiplication to **operands** such as variables and literal values. They usually return a new value that is the result of the operation and that can be assigned to a variable.

Most operators are binary, meaning that they work on two operands, as shown in the following pseudocode:

```
var resultOfOperation = firstOperand operator secondOperand;
```

Examples of binary operators include adding and multiplying, as shown in the following code:

```
int x = 5;
int y = 3;
int resultOfAdding = x + y;
int resultOfMultiplying = x * y;
```

Some operators are unary, meaning they work on a single operand, and can apply before or after the operand, as shown in the following pseudocode:

```
var resultOfOperationAfter = onlyOperand operator;
var resultOfOperationBefore = operator onlyOperand;
```

Examples of unary operators include incrementors and retrieving a type or its size in bytes, as shown in the following code:

```
int x = 5;
int postfixIncrement = x++;
int prefixIncrement = ++x;
Type theTypeOfAnInteger = typeof(int);
string nameOfVariable = nameof(x);
int howManyBytesInAnInteger = sizeof(int);
```

A ternary operator works on three operands, as shown in the following pseudocode:

```
var resultOfOperation = firstOperand firstOperator
  secondOperand secondOperator thirdOperand;
```

Exploring unary operators

Two common unary operators are used to increment, `++`, and decrement, `--`, a number. Let us write some example code to show how they work:

1. If you've completed the previous chapters, then you will already have a `cs11dotnet7` folder. If not, then you'll need to create it.
2. Use your preferred coding tool to create a new project, as defined in the following list:
   - Project template: **Console App** / `console`
   - Project file and folder: `Operators`
   - Workspace/solution file and folder: `Chapter03`
3. Open `Operators.csproj`, and after the `<PropertyGroup>` section, add a new `<ItemGroup>` section to statically import `System.Console` for all C# files using the implicit usings .NET SDK feature, as shown in the following markup:

```
<ItemGroup>
  <Using Include="System.Console" Static="true" />
</ItemGroup>
```

1. In `Program.cs`, delete the existing statements and then declare two integer variables named `a` and `b`, set `a` to `3`, increment `a` while assigning the result to `b`, and then output their values, as shown in the following code:

```
int a = 3;
int b = a++;
WriteLine($"a is {a}, b is {b}");
```

1. Before running the console app, ask yourself a question: what do you think the value of `b` will be when output? Once you've thought about that, run the code, and compare your prediction against the actual result, as shown in the following output:

```
a is 4, b is 3
```

The variable `b` has the value `3` because the `++` operator executes *after* the assignment; this is known as a **postfix operator**. If you need to increment *before* the assignment, then use the **prefix operator**.

1. Copy and paste the statements, and then modify them to rename the variables and use the prefix operator, as shown in the following code:

```
int c = 3;
int d = ++c; // increment c before assigning it
WriteLine($"c is {c}, d is {d}");
```

1. Rerun the code and note the result, as shown in the following output:

```
a is 4, b is 3
c is 4, d is 4
```

Good Practice: Due to the confusion between prefix and postfix for the increment and decrement operators when combined with an assignment, the Swift programming language designers decided to drop support for this operator in version 3. My recommendation for usage in C# is to never combine the use of `++` and `--` operators with an assignment operator, `=`. Perform the operations as separate statements.

Exploring binary arithmetic operators

Increment and decrement are unary arithmetic operators. Other arithmetic operators are usually binary and allow you to perform arithmetic operations on two numbers, as the following shows:

1. Add statements to declare and assign values to two integer variables named `e` and `f`, and then apply the five common binary arithmetic operators to the two numbers, as shown in the following code:

```
int e = 11;
int f = 3;
WriteLine($"e is {e}, f is {f}");
WriteLine($"e + f = {e + f}");
WriteLine($"e - f = {e - f}");
WriteLine($"e * f = {e * f}");
WriteLine($"e / f = {e / f}");
WriteLine($"e % f = {e % f}");
```

1. Run the code and note the result, as shown in the following output:

```
e is 11, f is 3
e + f = 14
e - f = 8
e * f = 33
e / f = 3
e % f = 2
```

To understand the divide `/` and modulo `%` operators when applied to integers, you need to think back to primary school. Imagine you have eleven sweets and three friends.

How can you divide the sweets between your friends? You can give three sweets to each of your friends, and there will be two left over. Those two sweets are the **modulus**, also known as the **remainder** after dividing. If you had twelve sweets, then each friend would get four of them, and there would be none left over, so the remainder would be 0.

1. Add statements to declare and assign a value to a `double` variable named `g` to show the difference between whole number and real number divisions, as shown in the following code:

```
double g = 11.0;
WriteLine($"g is {g:N1}, f is {f}");
WriteLine($"g / f = {g / f}");
```

1. Run the code and note the result, as shown in the following output:

```
g is 11.0, f is 3
g / f = 3.6666666666666665
```

If the first operand is a floating-point number, such as `g` with the value `11.0`, then the divide operator returns a floating-point value, such as `3.6666666666665`, rather than a whole number.

Assignment operators

You have already been using the most common assignment operator, `=`.

To make your code more concise, you can combine the assignment operator with other operators like arithmetic operators, as shown in the following code:

```
int p = 6;
p += 3; // equivalent to p = p + 3;
p -= 3; // equivalent to p = p - 3;
p *= 3; // equivalent to p = p * 3;
p /= 3; // equivalent to p = p / 3;
```

Exploring logical operators

Logical operators operate on Boolean values, so they return either `true` or `false`. Let's explore binary logical operators that operate on two Boolean values:

1. Use your preferred coding tool to add a new **Console App** / `console` project named `BooleanOperators` to the `Chapter03` workspace/solution.

   - In Visual Studio Code, select `BooleanOperators` as the active OmniSharp project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.
   - In Visual Studio 2022, set the startup project for the solution to the current selection.

   Good Practice: Remember to statically import `System.Console` for all C# files using the implicit usings .NET SDK feature.

2. In `Program.cs`, delete the existing statements and then add statements to declare two Boolean variables with values of `true` and `false`, and then output truth tables showing the results of applying AND, OR, and XOR (exclusive OR) logical operators, as shown in the following code:

```
bool a = true;
bool b = false;
WriteLine($"AND  | a     | b    ");
WriteLine($"a    | {a & a,-5} | {a & b,-5} ");
WriteLine($"b    | {b & a,-5} | {b & b,-5} ");
WriteLine();
WriteLine($"OR   | a     | b    ");
WriteLine($"a    | {a | a,-5} | {a | b,-5} ");
WriteLine($"b    | {b | a,-5} | {b | b,-5} ");
WriteLine();
WriteLine($"XOR  | a     | b    ");
WriteLine($"a    | {a ^ a,-5} | {a ^ b,-5} ");
WriteLine($"b    | {b ^ a,-5} | {b ^ b,-5} ");
```

1. Run the code and note the results, as shown in the following output:

```
AND  | a     | b
a    | True  | False
b    | False | False

OR   | a     | b
a    | True  | True
b    | True  | False
XOR  | a     | b
a    | False | True
b    | True  | False
```

For the AND `&` logical operator, both operands must be `true` for the result to be `true`. For the OR `|` logical operator, either operand can be `true` for the result to be `true`. For the XOR `^` logical operator, either operand can be `true` (but not both!) for the result to be `true`.

Exploring conditional logical operators

Conditional logical operators are like logical operators, but you use two symbols instead of one, for example, `&&` instead of `&`, or `||` instead of `|`.

In *Chapter 4*, *Writing, Debugging, and Testing Functions*, you will learn about functions in more detail, but I need to introduce functions now to explain conditional logical operators, also known as short-circuiting Boolean operators.

A function executes statements and then returns a value. That value could be a Boolean value like `true` that is used in a Boolean operation. Let's make use of conditional logical operators:

1. At the bottom of `Program.cs`, write statements to declare a function that writes a message to the console and returns `true`, as shown in the following code:

```
static bool DoStuff()
{
  WriteLine("I am doing some stuff.");
  return true;
}
```

Good Practice: If you are using a .NET Interactive Notebook, write the `DoStuff` function in a separate code cell and then execute it to make its context available to other code cells.

1. After the previous `WriteLine` statements, perform an AND `&` operation on the `a` and `b` variables, which you defined in *step 2* of the previous section, and the result of calling the function, as shown in the following code:

```
WriteLine();
WriteLine($"a & DoStuff() = {a & DoStuff()}");
WriteLine($"b & DoStuff() = {b & DoStuff()}");
```

1. Run the code, view the result, and note that the function was called twice, once for `a` and once for `b`, as shown in the following output:

```
I am doing some stuff.
a & DoStuff() = True
I am doing some stuff.
b & DoStuff() = False
```

1. Copy and paste the three statements and then change the `&` operators into `&&` operators, as shown in the following code:

```
WriteLine();
WriteLine($"a && DoStuff() = {a && DoStuff()}");
WriteLine($"b && DoStuff() = {b && DoStuff()}");
```

1. Run the code, view the result, and note that the function does run when combined with the `a` variable. It does not run when combined with the `b` variable because the `b` variable is `false` so the result will be `false` anyway, so it does not need to execute the function, as shown in the following output:

```
I am doing some stuff.
a && DoStuff() = True
b && DoStuff() = False // DoStuff function was not executed!
```

Good Practice: Now you can see why the conditional logical operators are described as being short-circuiting. They can make your apps more efficient, but they can also introduce subtle bugs in cases where you assume that the function will always be called. It is safest to avoid them when used in combination with functions that cause side effects.

Exploring bitwise and binary shift operators

Bitwise operators affect the bits in a number. Binary shift operators can perform some common arithmetic calculations much faster than traditional operators, for example, any multiplication by a factor of 2.

Let's explore bitwise and binary shift operators:

1. Use your preferred coding tool to add a new **Console App** / `console` project named `BitwiseAndShiftOperators` to the `Chapter03` workspace/solution.
   - In Visual Studio Code, select `BitwiseAndShiftOperators` as the active OmniSharp project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.
2. In `Program.cs`, delete the existing statements and then type statements to declare two integer variables with values 10 and 6, and then output the results of applying AND, OR, and XOR bitwise operators, as shown in the following code:

```
int a = 10; // 00001010
int b = 6;  // 00000110
```

```
WriteLine($"a = {a}");
WriteLine($"b = {b}");
WriteLine($"a & b = {a & b}"); // 2-bit column only e.g. 00000010
WriteLine($"a | b = {a | b}"); // 8, 4, and 2-bit columns e.g. 00001110
WriteLine($"a ^ b = {a ^ b}"); // 8 and 4-bit columns e.g. 00001100
```

1. Run the code and note the results, as shown in the following output:

```
a = 10
b = 6
a & b = 2
a | b = 14
a ^ b = 12
```

1. In `Program.cs`, add statements to output the results of applying the left-shift operator to move the bits of the variable `a` by three columns, multiplying `a` by 8, and right-shifting the bits of the variable `b` by one column, as shown in the following code:

```
// 01010000 left-shift a by three bit columns
WriteLine($"a << 3 = {a << 3}");
// multiply a by 8
WriteLine($"a * 8 = {a * 8}");
// 00000011 right-shift b by one bit column
WriteLine($"b >> 1 = {b >> 1}");
```

1. Run the code and note the results, as shown in the following output:

```
a << 3 = 80
a * 8 = 80
b >> 1 = 3
```

The `80` result is because the bits in it were shifted three columns to the left, so the 1-bits moved into the 64- and 16-bit columns and 64 + 16 = 80. This is the equivalent of multiplying by 8, but CPUs can perform a bit-shift faster. The 3 result is because the 1-bits in `b` were shifted one column into the 2- and 1-bit columns.

Good Practice: Remember that when operating on integer values, the `&` and `|` symbols are bitwise operators, and when operating on Boolean values like `true` and `false`, the `&` and `|` symbols are logical operators.

We can illustrate the operations by converting the integer values into binary strings of zeros and ones:

1. At the bottom of `Program.cs`, add a function to convert an integer value into a binary (Base2) `string` of up to eight zeros and ones, as shown in the following code:

```
static string ToBinaryString(int value)
{
  return Convert.ToString(value, toBase: 2).PadLeft(8, '0');
}
```

1. Above the function, add statements to output `a`, `b`, and the results of the various bitwise operators, as shown in the following code:

```
WriteLine();
WriteLine("Outputting integers as binary:");
WriteLine($"a =     {ToBinaryString(a)}");
WriteLine($"b =     {ToBinaryString(b)}");
WriteLine($"a & b = {ToBinaryString(a & b)}");
WriteLine($"a | b = {ToBinaryString(a | b)}");
WriteLine($"a ^ b = {ToBinaryString(a ^ b)}");
```

1. Run the code and note the results, as shown in the following output:

```
Outputting integers as binary:
a =     00001010
b =     00000110
```

```
a & b = 00000010
a | b = 00001110
a ^ b = 00001100
```

Miscellaneous operators

`nameof` and `sizeof` are convenient operators when working with types:

- `nameof` returns the short name (without the namespace) of a variable, type, or member as a `string` value, which is useful when outputting exception messages.
- `sizeof` returns the size in bytes of simple types, which is useful for determining the efficiency of data storage.

For example:

```
int age = 50;
WriteLine($"The {nameof(age)} variable uses {sizeof(int)} bytes of memory.");
```

There are many other operators; for example, the dot between a variable and its members is called the **member access operator** and the round brackets at the end of a function or method name are called the **invocation operator**, as shown in the following code:

```
int age = 50;
// How many operators in the following statement?
char firstDigit = age.ToString()[0];
// There are four operators:
// = is the assignment operator
// . is the member access operator
// () is the invocation operator
// [] is the indexer access operator
```

Understanding selection statements

Every application needs to be able to select from choices and branch along different code paths. The two selection statements in C# are `if` and `switch`. You can use `if` for all your code, but `switch` can simplify your code in some common scenarios, such as when there is a single variable that can have multiple values that each requires different processing.

Branching with the if statement

The `if` statement determines which branch to follow by evaluating a Boolean expression. If the expression is `true`, then the block executes. The `else` block is optional, and it executes if the `if` expression is `false`. The `if` statement can be nested.

The `if` statement can be combined with other `if` statements as `else if` branches, as shown in the following code:

```
if (expression1)
{
  // runs if expression1 is true
}
else if (expression2)
{
  // runs if expression1 is false and expression2 if true
}
else if (expression3)
{
  // runs if expression1 and expression2 are false
  // and expression3 is true
}
else
{
```

```
  // runs if all expressions are false
}
```

Each `if` statement's Boolean expression is independent of the others and, unlike `switch` statements, does not need to reference a single value.

Let's write some code to explore selection statements like `if`:

1. Use your preferred coding tool to add a new **Console App** / `console` project named `SelectionStatements` to the `Chapter03` workspace/solution.
   - In Visual Studio Code, select `SelectionStatements` as the active OmniSharp project.
2. In `Program.cs`, delete the existing statements and then add statements to check if a password is at least eight characters, as shown in the following code:

```
string password = "ninja";
if (password.Length < 8)
{
  WriteLine("Your password is too short. Use at least 8 characters.");
}
else
{
  WriteLine("Your password is strong.");
}
```

1. Run the code and note the result, as shown in the following output:

```
 Your password is too short. Use at least 8 characters.
```

Why you should always use braces with if statements

As there is only a single statement inside each block, the preceding code could be written without the curly braces, as shown in the following code:

```
if (password.Length < 8)
  WriteLine("Your password is too short. Use at least 8 characters.");
else
  WriteLine("Your password is strong.");
```

This style of `if` statement should be avoided because it can introduce serious bugs, for example, the infamous #gotofail bug in Apple's iPhone iOS operating system.

For 18 months after Apple's iOS 6 was released, in September 2012, it had a bug in its **Secure Sockets Layer** (**SSL**) encryption code, which meant that any user running Safari, the device's web browser, who tried to connect to secure websites, such as their bank, was not properly secure because an important check was being accidentally skipped.

Just because you can leave out the curly braces doesn't mean you should. Your code is not "more efficient" without them; instead, it is less maintainable and potentially more dangerous.

Pattern matching with the if statement

A feature introduced with C# 7.0 and later is pattern matching. The `if` statement can use the `is` keyword in combination with declaring a local variable to make your code safer:

1. Add statements so that if the value stored in the variable named `o` is an `int`, then the value is assigned to the local variable named `i`, which can then be used inside the `if` statement. This is safer than using the variable named `o` because we know for sure that `i` is an `int` variable and not something else, as shown in the following code:

```
// add and remove the "" to change the behavior
object o = "3";
```

```
int j = 4;
if (o is int i)
{
  WriteLine($"{i} x {j} = {i * j}");
}
else
{
  WriteLine("o is not an int so it cannot multiply!");
}
```

1. Run the code and view the results, as shown in the following output:

```
o is not an int so it cannot multiply!
```

1. Delete the double-quote characters around the `"3"` value so that the value stored in the variable named `o` is an `int` type instead of a `string` type.
2. Rerun the code to view the results, as shown in the following output:

```
3 x 4 = 12
```

Branching with the switch statement

The `switch` statement is different from the `if` statement because `switch` compares a single expression against a list of multiple possible `case` statements. Every `case` statement is related to the single expression. Every `case` section must end with:

- The `break` keyword (like case 1 in the following code)
- Or the `goto case` keywords (like case 2 in the following code)
- Or they should have no statements (like case 3 in the following code)
- Or the `goto` keyword that references a named label (like case 5 in the following code)
- Or the `return` keyword to leave the current function (not shown in the code)

Let's write some code to explore the `switch` statements:

1. Type code for a `switch` statement. You should note that the penultimate statement is a label that can be jumped to, and the first statement generates a random number between 1 and 6 (the number 7 in the code is an exclusive upper bound). The `switch` statement branches are based on the value of this random number, as shown in the following code:

```
int number = Random.Shared.Next(1, 7);
WriteLine($"My random number is {number}");
switch (number)
{
  case 1:
    WriteLine("One");
    break; // jumps to end of switch statement
  case 2:
    WriteLine("Two");
    goto case 1;
  case 3: // multiple case section
  case 4:
    WriteLine("Three or four");
    goto case 1;
  case 5:
    goto A_label;
  default:
    WriteLine("Default");
    break;
} // end of switch statement
WriteLine("After end of switch");
A_label:
WriteLine($"After A_label");
```

Good Practice: You can use the `goto` keyword to jump to another case or a label. The `goto` keyword is frowned upon by most programmers but can be a good solution to code logic in some scenarios. However, you should use it sparingly.

1. Run the code multiple times to see what happens in various cases of random numbers, as shown in the following example output:

```
// first random run
My random number is 4
Three or four
One
After end of switch
After A_label
// second random run
My random number is 2
Two
One
After end of switch
After A_label
// third random run
My random number is 6
Default
After end of switch
After A_label
// fourth random run
My random number is 1
One
After end of switch
After A_label
// fifth random run
My random number is 5
After A_label
```

Good Practice: The `Random` class that we used to generate a random number has a `Next` method that allows you to specify an inclusive lower bound and an exclusive upper bound and will generate a pseudo-random number. Instead of creating a new instance of `Random` that is not thread-safe, since .NET 6 you can use a `Shared` instance that is thread-safe so it can be used concurrently from any thread.

Pattern matching with the switch statement

Like the `if` statement, the `switch` statement supports pattern matching in C# 7.0 and later. The `case` values no longer need to be literal values; they can be patterns.

Let's see an example of pattern matching with the `switch` statement using a folder path. If you are using macOS, then swap in the commented statement that sets the path variable and replace my username with your user folder name:

1. Add statements to declare a `string` path to a file, open it as either a read-only or writeable stream, and then show a message based on what type and capabilities the stream has, as shown in the following code:

```
// string path = "/Users/markjprice/cs11dotnet7/Chapter03";
string path = @"C:\cs11dotnet7\Chapter03";
Stream? s;
Write("Press R for read-only or W for writeable: ");
ConsoleKeyInfo key = ReadKey();
if (key.Key == ConsoleKey.R)
{
  s =  File.Open(
    Path.Combine(path, "file.txt"),
    FileMode.OpenOrCreate,
    FileAccess.Read);
}
else
{
  s =  File.Open(
```

```
    Path.Combine(path, "file.txt"),
    FileMode.OpenOrCreate,
    FileAccess.Write);
}
WriteLine();
string message;
switch (s)
{
  case FileStream writeableFile when s.CanWrite:
    message = "The stream is a file that I can write to.";
    break;
  case FileStream readOnlyFile:
    message = "The stream is a read-only file.";
    break;
  case MemoryStream ms:
    message = "The stream is a memory address.";
    break;
  default: // always evaluated last despite its current position
    message = "The stream is some other type.";
    break;
  case null:
    message = "The stream is null.";
    break;
}
WriteLine(message);
```

1. Run the code and note that the variable named `s` is declared as a `Stream` type so it could be any subtype of stream, such as a memory stream or file stream. In this code, the stream is created using the `File.Open` method, which returns a file stream and, depending on your key press, it will be writeable or read-only, so the result will be a message that describes the situation, as shown in the following output:

```
The stream is a file that I can write to.
```

In .NET, there are multiple subtypes of `Stream`, including `FileStream` and `MemoryStream`. In C# 7.0 and later, your code can more concisely branch, based on the subtype of stream, and declare and assign a local variable to safely use it. You will learn more about the `System.IO` namespace and the `Stream` type in *Chapter 9*, *Working with Files, Streams, and Serialization*.

Additionally, `case` statements can include a `when` keyword to perform more specific pattern matching. In the first case statement in the preceding code, `s` will only be a match if the stream is a `FileStream` and its `CanWrite` property is `true`.

Simplifying switch statements with switch expressions

In C# 8.0 or later, you can simplify `switch` statements using **switch expressions**.

Most `switch` statements are very simple, yet they require a lot of typing. `switch` expressions are designed to simplify the code you need to type while still expressing the same intent in scenarios where all cases return a value to set a single variable. `switch` expressions use a lambda, `=>`, to indicate a return value.

Let's implement the previous code that used a `switch` statement using a `switch` expression so that you can compare the two styles:

1. At the bottom of `Program.cs`, add statements to set the message based on what type and capabilities the stream has, using a `switch` expression, as shown in the following code:

```
message = s switch
{
  FileStream writeableFile when s.CanWrite
    => "The stream is a file that I can write to.",
  FileStream readOnlyFile
    => "The stream is a read-only file.",
  MemoryStream ms
    => "The stream is a memory address.",
```

```
  null
    => "The stream is null.",

  _
    => "The stream is some other type."
};
WriteLine(message);
```

The main differences are the removal of the `case` and `break` keywords. The underscore character `_` is used to represent the default return value.

1. Run the code, and note that the result is the same as before.

Understanding iteration statements

Iteration statements repeat a block of statements either while a condition is `true` (`while` and `for` statements) or for each item in a collection (`foreach` statement). The choice of which statement to use is based on a combination of ease of understanding to solve the logic problem and personal preference.

Looping with the while statement

The `while` statement evaluates a Boolean expression and continues to loop while it is true. Let's explore iteration statements:

1. Use your preferred coding tool to add a new **Console App** / `console` project named `IterationStatements` to the `Chapter03` workspace/solution.
   - In Visual Studio Code, select `IterationStatements` as the active OmniSharp project.
2. In `Program.cs`, delete the existing statements and then add statements to define a `while` statement that loops while an integer variable has a value less than 10, as shown in the following code:

```
int x = 0;
while (x < 10)
{
  WriteLine(x);
  x++;
}
```

1. Run the code and view the results, which should be the numbers 0 to 9, as shown in the following output:

```
0
1
2
3
4
5
6
7
8
9
```

Looping with the do statement

The `do` statement is like `while`, except the Boolean expression is checked at the bottom of the block instead of the top, which means that the block always executes at least once, as the following shows:

1. Type statements to define a `do` loop, as shown in the following code:

```
string? password;
do
{
  Write("Enter your password: ");
  password = ReadLine();
}
```

```
while (password != "Pa$$w0rd");
WriteLine("Correct!");
```

1. Run the code, and note that you are prompted to enter your password repeatedly until you enter it correctly, as shown in the following output:

```
Enter your password: password
Enter your password: 12345678
Enter your password: ninja
Enter your password: correct horse battery staple
Enter your password: Pa$$w0rd
Correct!
```

1. As an optional challenge, add statements so that the user can only make ten attempts before an error message is displayed.

Looping with the for statement

The `for` statement is like `while`, except that it is more succinct. It combines:

- An optional **initializer expression**, which executes once at the start of the loop.
- An optional **conditional expression**, which executes on every iteration at the start of the loop to check whether the looping should continue. If the expression returns `true` or it is missing, the loop will execute again.
- An optional **iterator expression**, which executes on every loop at the bottom of the statement. This is often used to increment a counter variable.

The `for` statement is commonly used with an integer counter. Let's explore some code:

1. Type a `for` statement to output the numbers 1 to 10, as shown in the following code:

```
for (int y = 1; y <= 10; y++)
{
  WriteLine(y);
}
```

1. Run the code to view the result, which should be the numbers 1 to 10.

Looping with the foreach statement

The `foreach` statement is a bit different from the previous three iteration statements.

It is used to perform a block of statements on each item in a sequence, for example, an array or collection. Each item is usually read-only, and if the sequence structure is modified during iteration, for example, by adding or removing an item, then an exception will be thrown.

Try the following example:

1. Type statements to create an array of string variables and then output the length of each one, as shown in the following code:

```
string[] names = { "Adam", "Barry", "Charlie" };
foreach (string name in names)
{
  WriteLine($"{name} has {name.Length} characters.");
}
```

1. Run the code and view the results, as shown in the following output:

```
Adam has 4 characters.
Barry has 5 characters.
Charlie has 7 characters.
```

Understanding how foreach works internally

A creator of any type that represents multiple items, like an array or collection, should make sure that a programmer can use the `foreach` statement to enumerate through the type's items.

Technically, the `foreach` statement will work on any type that follows these rules:

- The type must have a method named `GetEnumerator` that returns an object.
- The returned object must have a property named `Current` and a method named `MoveNext`.
- The `MoveNext` method must change the value of `Current` and return `true` if there are more items to enumerate through or return `false` if there are no more items.

There are interfaces named `IEnumerable` and `IEnumerable<T>` that formally define these rules, but technically the compiler does not require the type to implement these interfaces.

The compiler turns the `foreach` statement in the preceding example into something like the following pseudocode:

```
IEnumerator e = names.GetEnumerator();
while (e.MoveNext())
{
  string name = (string)e.Current; // Current is read-only!
  WriteLine($"{name} has {name.Length} characters.");
}
```

Due to the use of an iterator and its read-only `Current` property, the variable declared in a `foreach` statement cannot be used to modify the value of the current item.

Storing multiple values in an array

When you need to store multiple values of the same type, you can declare an **array**. For example, you may do this when you need to store four names in a `string` array.

Working with single-dimensional arrays

The code that you will write next will allocate memory for an array for storing four `string` values. It will then store `string` values at index positions 0 to 3 (arrays usually have a lower bound of zero, so the index of the last item is one less than the length of the array).

We could visualize the array like this:

0	1	2	3
Kate	Jack	Rebecca	Tom

> **Good Practice**: Do not assume that all arrays count from zero. The most common type of array in .NET is an **szArray**, a single-dimension zero-indexed array, and these use the normal `[]` syntax. But .NET also has **mdArray**, a multi-dimensional array, and they do not have to have a lower bound of zero. These are rarely used, but you should know they exist.

Finally, it will loop through each item in the array using a `for` statement.

Let's look at how to use an array:

1. Use your preferred code editor to add a new **Console App** / `console` project named `Arrays` to the `Chapter03` workspace/solution.
   - In Visual Studio Code, select `Arrays` as the active OmniSharp project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.

2. In `Program.cs`, delete the existing statements and then type statements to declare and use an array of `string` values, as shown in the following code:

```
string[] names; // can reference any size array of strings
// allocating memory for four strings in an array
names = new string[4];
// storing items at index positions
names[0] = "Kate";
names[1] = "Jack";
names[2] = "Rebecca";
names[3] = "Tom";
// looping through the names
for (int i = 0; i < names.Length; i++)
{
  // output the item at index position i
  WriteLine(names[i]);
}
```

1. Run the code and note the result, as shown in the following output:

```
Kate
Jack
Rebecca
Tom
```

Arrays are always of a fixed size at the time of memory allocation, so you need to decide how many items you want to store before instantiating them.

An alternative to defining the array in three steps as above is to use array initializer syntax.

1. Before the `for` loop, add a statement to declare, allocate memory, and instantiate the values of a similar array, as shown in the following code:

```
string[] names2 = new[] { "Kate", "Jack", "Rebecca", "Tom" };
```

1. Change the `for` loop to use `names2`, run the console app, and note that the results are the same.

When you use the `new[]` syntax to allocate memory for the array, you must have at least one item in the curly braces so that the compiler can infer the data type.

Working with multi-dimensional arrays

Instead of a single-dimension array for storing a row of string values (or any other data type), what if we want to store a grid of values? Or a cube? Or even higher dimensions?

We could visualize a two-dimensional array, aka grid, of `string` values like this:

	0	1	2	3
0	Alpha	Beta	Gamma	Delta
1	Anne	Ben	Charlie	Doug
2	Aardvark	Bear	Cat	Dog

Let's look at how to use multi-dimensional arrays:

1. At the bottom of `Program.cs`, add statements to declare and instantiate a two-dimensional array of `string` values, as shown in the following code:

```
string[,] grid1 = new[,] // two dimensions
{
  { "Alpha", "Beta", "Gamma", "Delta" },
  { "Anne", "Ben", "Charlie", "Doug" },
```

```
  { "Aardvark", "Bear", "Cat", "Dog" }
};
```

1. We can discover the lower and upper bounds of this array using helpful methods, as shown in the following code:

```
WriteLine($"Lower bound of the first dimension is: {grid1.GetLowerBound(0)}");
WriteLine($"Upper bound of the first dimension is: {grid1.GetUpperBound(0)}");
WriteLine($"Lower bound of the second dimension is: {grid1.GetLowerBound(1)}");
WriteLine($"Upper bound of the second dimension is: {grid1.GetUpperBound(1)}");
```

1. Run the code and note the result, as shown in the following output:

```
Lower bound of the first dimension is: 0
Upper bound of the first dimension is: 2
Lower bound of the second dimension is: 0
Upper bound of the second dimension is: 3
```

1. We can then use these values in nested `for` statements to loop through the `string` values, as shown in the following code:

```
for (int row = 0; row <= grid1.GetUpperBound(0); row++)
{
  for (int col = 0; col <= grid1.GetUpperBound(1); col++)
  {
    WriteLine($"Row {row}, Column {col}: {grid1[row, col]}");
  }
}
```

1. Run the code and note the result, as shown in the following output:

```
Row 0, Column 0: Alpha
Row 0, Column 1: Beta
Row 0, Column 2: Gamma
Row 0, Column 3: Delta
Row 1, Column 0: Anne
Row 1, Column 1: Ben
Row 1, Column 2: Charlie
Row 1, Column 3: Doug
Row 2, Column 0: Aardvark
Row 2, Column 1: Bear
Row 2, Column 2: Cat
Row 2, Column 3: Dog
```

You must supply a value for every row and every column when it is instantiated, or you will get compile errors. If you need to indicate a missing `string` value, then use `string.Empty`. Or if you declare the array to be nullable `string` values by using `string?[]`, then you can also use `null` for a missing value.

If you cannot use the array initialization syntax, perhaps because you are loading values from a file or database, then you can separate the declaration of the array dimension and the allocation of memory from the assigning of values, as shown in the following code:

```
// alternative syntax
string[,] grid2 = new string[3,4]; // allocate memory
grid2[0, 0] = "Alpha"; // assign values
grid2[0, 1] = "Beta";
// and so on
grid2[2, 3] = "Dog";
```

When declaring the size of the dimensions, you specify the length, not the upper bound. The expression `new string[3,4]` means the array can have 3 items in its first dimension (0) with an upper bound of 2, and the array can have 4 items in its second dimension (1) with an upper bound of 3.

Working with jagged arrays

If you need a multi-dimensional array but the number of items stored in each dimension are different, then you can define an array of arrays, aka a jagged array.

We could visualize a jagged array like this:

0	0	1	2	
	Alpha	Beta	Gamma	
1	0	1	2	3
	Anne	Ben	Charlie	Doug
2	0	1		
	Aardvark	Bear		

Let's look at how to use a jagged array:

1. At the bottom of `Program.cs`, add statements to declare and instantiate an array of arrays of `string` values, as shown in the following code:

```
string[][] jagged = new[] // array of string arrays
{
  new[] { "Alpha", "Beta", "Gamma" },
  new[] { "Anne", "Ben", "Charlie", "Doug" },
  new[] { "Aardvark", "Bear" }
};
```

1. We can discover the lower and upper bounds of the array of arrays, and then each array with it, as shown in the following code:

```
WriteLine("Upper bound of array of arrays is: {0}",
  jagged.GetUpperBound(0));
for (int array = 0; array <= jagged.GetUpperBound(0); array++)
{
  WriteLine("Upper bound of array {0} is: {1}",
    arg0: array,
    arg1: jagged[array].GetUpperBound(0));
}
```

1. Run the code and note the result, as shown in the following output:

```
Upper bound of array of arrays is: 2
Upper bound of array 0 is: 2
Upper bound of array 1 is: 3
Upper bound of array 2 is: 1
```

1. We can then use these values in nested `for` statements to loop through the `string` values, as shown in the following code:

```
for (int row = 0; row <= jagged.GetUpperBound(0); row++)
{
  for (int col = 0; col <= jagged[row].GetUpperBound(0); col++)
  {
    WriteLine($"Row {row}, Column {col}: {jagged[row][col]}");
  }
}
```

1. Run the code and note the result, as shown in the following output:

```
Row 0, Column 0: Alpha
Row 0, Column 1: Beta
Row 0, Column 2: Gamma
Row 1, Column 0: Anne
Row 1, Column 1: Ben
Row 1, Column 2: Charlie
Row 1, Column 3: Doug
```

```
Row 2, Column 0: Aardvark
Row 2, Column 1: Bear
```

Summarizing arrays

We use slightly different syntax to declare different types of arrays, as shown in the following table:

Type of array	Declaration syntax
Single dimension	`datatype[]` , for example, `string[]`
Two dimensions	`string[,]`
Three dimensions	`string[,,]`
Ten dimensions	`string[,,,,,,,,,]`
Array of arrays aka jagged array	`string[][]`
Array of arrays of arrays	`string[][][]`

Arrays are useful for temporarily storing multiple items, but collections are a more flexible option when adding and removing items dynamically. You don't need to worry about collections right now, as we will cover them in *Chapter 8*, *Working with Common .NET Types*.

Casting and converting between types

You will often need to convert values of variables between different types. For example, data input is often entered as text at the console, so it is initially stored in a variable of the `string` type, but it then needs to be converted into a date/time, or number, or some other data type, depending on how it should be stored and processed.

Sometimes you will need to convert between number types, like between an integer and a floating point, before performing calculations.

Converting is also known as **casting**, and it has two varieties: **implicit** and **explicit**. Implicit casting happens automatically, and it is safe, meaning that you will not lose any information.

Explicit casting must be performed manually because it may lose information, for example, the precision of a number. By explicitly casting, you are telling the C# compiler that you understand and accept the risk.

Casting numbers implicitly and explicitly

Implicitly casting an `int` variable into a `double` variable is safe because no information can be lost, as the following shows:

1. Use your preferred coding tool to add a new **Console App** / `console` project named `CastingConverting` to the `Chapter03` workspace/solution.
   - In Visual Studio Code, select `CastingConverting` as the active OmniSharp project.
2. In `Program.cs` , delete the existing statements and then type statements to declare and assign an `int` variable and a `double` variable, and then implicitly cast the integer's value when assigning it to the `double` variable, as shown in the following code:

```
int a = 10;
double b = a; // an int can be safely cast into a double
WriteLine(b);
```

1. Type statements to declare and assign a `double` variable and an `int` variable, and then implicitly cast the `double` value when assigning it to the `int` variable, as shown in the following code:

```
double c = 9.8;
int d = c; // compiler gives an error for this line
WriteLine(d);
```

1. Run the code and note the error message, as shown in the following output:

```
Error: (6,9): error CS0266: Cannot implicitly convert type 'double' to 'int'. An explicit convers
```

This error message will also appear in the Visual Studio **Error List** or Visual Studio Code **PROBLEMS** window.

You cannot implicitly cast a `double` variable into an `int` variable because it is potentially unsafe and could lose data, like the value after the decimal point. You must explicitly cast a `double` variable into an `int` variable using a pair of round brackets around the type you want to cast the `double` type into. The pair of round brackets is the **cast operator**. Even then, you must beware that the part after the decimal point will be trimmed off without warning because you have chosen to perform an explicit cast and therefore understand the consequences.

1. Modify the assignment statement for the `d` variable, as shown in the following code:

```
int d = (int)c;
WriteLine(d); // d is 9 losing the .8 part
```

1. Run the code to view the results, as shown in the following output:

```
10
9
```

We must perform a similar operation when converting values between larger integers and smaller integers. Again, beware that you might lose information because any value too big will have its bits copied and then be interpreted in ways that you might not expect!

1. Enter statements to declare and assign a long 64-bit variable to an int 32-bit variable, both using a small value that will work and a too-large value that will not, as shown in the following code:

```
long e = 10;
int f = (int)e;
WriteLine($"e is {e:N0} and f is {f:N0}");
e = long.MaxValue;
f = (int)e;
WriteLine($"e is {e:N0} and f is {f:N0}");
```

1. Run the code to view the results, as shown in the following output:

```
e is 10 and f is 10
e is 9,223,372,036,854,775,807 and f is -1
```

1. Modify the value of `e` to 5 billion, as shown in the following code:

```
e = 5_000_000_000;
```

1. Run the code to view the results, as shown in the following output:

```
e is 5,000,000,000 and f is 705,032,704
```

Converting with the System.Convert type

You can only cast between similar types, for example, between whole numbers like `byte`, `int`, and `long`, or between a class and its subclasses. You cannot cast a `long` to a `string` or a `byte` to a `DateTime`.

An alternative to using the cast operator is to use the `System.Convert` type. The `System.Convert` type can convert to and from all the C# number types, as well as Booleans, strings, and date and time values.

Let's write some code to see this in action:

1. At the top of `Program.cs`, statically import the `System.Convert` class, as shown in the following code:

```
using static System.Convert;
```

1. At the bottom of `Program.cs`, type statements to declare and assign a value to a `double` variable, convert it to an integer, and then write both values to the console, as shown in the following code:

```
double g = 9.8;
int h = ToInt32(g); // a method of System.Convert
WriteLine($"g is {g} and h is {h}");
```

1. Run the code and view the result, as shown in the following output:

```
g is 9.8 and h is 10
```

An important difference between casting and converting is that converting rounds the `double` value `9.8` up to `10` instead of trimming the part after the decimal point.

Rounding numbers

You have now seen that the cast operator trims the decimal part of a real number and that the `System.Convert` methods round up or down. However, what is the rule for rounding?

Understanding the default rounding rules

In British primary schools for children aged 5 to 11, pupils are taught to round *up* if the decimal part is .5 or higher and round *down* if the decimal part is less.

Let's explore if C# follows the same primary school rule:

1. Type statements to declare and assign an array of `double` values, convert each of them to an integer, and then write the result to the console, as shown in the following code:

```
double[] doubles = new[]
  { 9.49, 9.5, 9.51, 10.49, 10.5, 10.51 };
foreach (double n in doubles)
{
  WriteLine($"ToInt32({n}) is {ToInt32(n)}");
}
```

1. Run the code and view the result, as shown in the following output:

```
ToInt32(9.49) is 9
ToInt32(9.5) is 10
ToInt32(9.51) is 10
ToInt32(10.49) is 10
ToInt32(10.5) is 10
ToInt32(10.51) is 11
```

We have shown that the rule for rounding in C# is subtly different from the primary school rule:

- It always rounds *down* if the decimal part is less than the midpoint .5.
- It always rounds *up* if the decimal part is more than the midpoint .5.
- It will round *up* if the decimal part is the midpoint .5 and the non-decimal part is *odd*, but it will round *down* if the non-decimal part is *even*.

This rule is known as **Banker's Rounding**, and it is preferred because it reduces bias by alternating when it rounds up or down. Sadly, other languages such as JavaScript use the primary school rule.

Taking control of rounding rules

You can take control of the rounding rules by using the `Round` method of the `Math` class:

1. Type statements to round each of the `double` values using the "away from zero" rounding rule, also known as rounding "up," and then write the result to the console, as shown in the following code:

```
foreach (double n in doubles)
{
  WriteLine(format:
    "Math.Round({0}, 0, MidpointRounding.AwayFromZero) is {1}",
    arg0: n,
    arg1: Math.Round(value: n, digits: 0,
            mode: MidpointRounding.AwayFromZero));
}
```

1. Run the code and view the result, as shown in the following output:

```
Math.Round(9.49, 0, MidpointRounding.AwayFromZero) is 9
Math.Round(9.5, 0, MidpointRounding.AwayFromZero) is 10
Math.Round(9.51, 0, MidpointRounding.AwayFromZero) is 10
Math.Round(10.49, 0, MidpointRounding.AwayFromZero) is 10
Math.Round(10.5, 0, MidpointRounding.AwayFromZero) is 11
Math.Round(10.51, 0, MidpointRounding.AwayFromZero) is 11
```

Good Practice: For every programming language that you use, check its rounding rules. They may not work the way you expect!

Converting from any type to a string

The most common conversion is from any type into a `string` variable for outputting as human-readable text, so all types have a method named `ToString` that they inherit from the `System.Object` class.

The `ToString` method converts the current value of any variable into a textual representation. Some types can't be sensibly represented as text, so they return their namespace and type name instead.

Let's convert some types into a `string`:

1. Type statements to declare some variables, convert them to their `string` representation, and write them to the console, as shown in the following code:

```
int number = 12;
WriteLine(number.ToString());
bool boolean = true;
WriteLine(boolean.ToString());
DateTime now = DateTime.Now;
WriteLine(now.ToString());
object me = new();
WriteLine(me.ToString());
```

1. Run the code and view the result, as shown in the following output:

```
12
True
08/28/2022 17:33:54
System.Object
```

Converting from a binary object to a string

When you have a binary object like an image or video that you want to either store or transmit, you sometimes do not want to send the raw bits because you do not know how those bits could be misinterpreted, for example, by the network protocol transmitting them or another operating system that is reading the stored binary object.

The safest thing to do is to convert the binary object into a `string` of safe characters. Programmers call this **Base64** encoding.

The `Convert` type has a pair of methods, `ToBase64String` and `FromBase64String`, that perform this conversion for you. Let's see them in action:

1. Type statements to create an array of bytes randomly populated with byte values, write each byte nicely formatted to the console, and then write the same bytes converted to Base64 to the console, as shown in the following code:

```
// allocate array of 128 bytes
byte[] binaryObject = new byte[128];
// populate array with random bytes
Random.Shared.NextBytes(binaryObject);
WriteLine("Binary Object as bytes:");
for(int index = 0; index < binaryObject.Length; index++)
{
  Write($"{binaryObject[index]:X} ");
}
WriteLine();
// convert to Base64 string and output as text
string encoded = ToBase64String(binaryObject);
WriteLine($"Binary Object as Base64: {encoded}");
```

By default, an `int` value would output assuming decimal notation, that is, base10. You can use format codes such as `:X` to format the value using hexadecimal notation.

1. Run the code and view the result, as shown in the following output:

```
Binary Object as bytes:
B3 4D 55 DE 2D E BB CF BE 4D E6 53 C3 C2 9B 67 3 45 F9 E5 20 61 7E 4F 7A 81
EC 49 F0 49 1D 8E D4 F7 DB 54 AF A0 81 5 B8 BE CE F8 36 90 7A D4 36 42
4 75 81 1B AB 51 CE 5 63 AC 22 72 DE 74 2F 57 7F CB E7 47 B7 62 C3 F4 2D
61 93 85 18 EA 6 17 12 AE 44 A8 D B8 4C 89 85 A9 3C D5 E2 46 E0 59 C9 DF
10 AF ED EF 8AA1 B1 8D EE 4A BE 48 EC 79 A5 A 5F 2F 30 87 4A C7 7F 5D C1 D
26 EE
Binary Object as Base64: s01V3i0Ou8++TeZTw8KbZwNF +eUgYX5PeoHsSfBJHY7U99tU r6CBBbi+zvg2kHrUNkIEd
```

Parsing from strings to numbers or dates and times

The second most common conversion is from strings to numbers or date and time values.

The opposite of `ToString` is `Parse`. Only a few types have a `Parse` method, including all the number types and `DateTime`.

Let's see `Parse` in action:

1. Type statements to parse an integer and a date and time value from strings and then write the result to the console, as shown in the following code:

```
int age = int.Parse("27");
DateTime birthday = DateTime.Parse("4 July 1980");
WriteLine($"I was born {age} years ago.");
WriteLine($"My birthday is {birthday}.");
WriteLine($"My birthday is {birthday:D}.");
```

1. Run the code and view the result, as shown in the following output:

```
I was born 27 years ago.
My birthday is 04/07/1980 00:00:00.
My birthday is 04 July 1980.
```

By default, a date and time value outputs with the short date and time format. You can use format codes such as `D` to output only the date part using the long date format.

Good Practice: Use the standard date and time format specifiers, as shown at the following link: https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-date-and-time-format-strings#table-of-

[format-specifiers]

Errors using Parse

One problem with the `Parse` method is that it gives errors if the `string` cannot be converted.

1. Type a statement to attempt to parse a string containing letters into an integer variable, as shown in the following code:

```
int count = int.Parse("abc");
```

1. Run the code and view the result, as shown in the following output:

```
Unhandled Exception: System.FormatException: Input string was not in a correct format.
```

As well as the preceding exception message, you will see a stack trace. I have not included stack traces in this book because they take up too much space.

Avoiding exceptions using the TryParse method

To avoid errors, you can use the `TryParse` method instead. `TryParse` attempts to convert the input `string` and returns `true` if it can convert it and `false` if it cannot. Exceptions are a relatively expensive operation so they should be avoided when you can.

The `out` keyword is required to allow the `TryParse` method to set the count variable when the conversion works.

Let's see `TryParse` in action:

1. Replace the `int count` declaration with statements to use the `TryParse` method and ask the user to input a count for a number of eggs, as shown in the following code:

```
Write("How many eggs are there? ");
string? input = ReadLine(); // or use "12" in notebook
if (int.TryParse(input, out int count))
{
  WriteLine($"There are {count} eggs.");
}
else
{
  WriteLine("I could not parse the input.");
}
```

1. Run the code, enter `12`, and view the result, as shown in the following output:

```
How many eggs are there? 12
There are 12 eggs.
```

1. Run the code, enter `twelve` (or change the `string` value to `"twelve"` in a notebook), and view the result, as shown in the following output:

```
How many eggs are there? twelve
I could not parse the input.
```

You can also use methods of the `System.Convert` type to convert `string` values into other types; however, like the `Parse` method, it gives an error if it cannot convert.

Handling exceptions

You've seen several scenarios where errors have occurred when converting types. Some languages return error codes when something goes wrong. .NET uses exceptions that are richer and designed only for failure reporting.

When this happens, we say *a runtime exception has been thrown*.

Other systems might use return values that could have multiple uses. For example, if the return value is a positive number, it might represent the count of rows in a table, or if the return value is a negative number it might represent some error code.

When an exception is thrown, the thread is suspended and if the calling code has defined a `try-catch` statement, then it is given a chance to handle the exception. If the current method does not handle it, then its calling method is given a chance, and so on up the call stack.

As you have seen, the default behavior of a console app or a .NET Interactive notebook is to output a message about the exception, including a stack trace, and then stop running the code. The application is terminated. This is better than allowing the code to continue executing in a potentially corrupt state. Your code should only catch and handle exceptions that it understands and can properly fix.

> **Good Practice**: Avoid writing code that will throw an exception whenever possible, perhaps by performing `if` statement checks. Sometimes you can't, and sometimes it is best to allow the exception to be caught by a higher-level component that is calling your code. You will learn how to do this in *Chapter 4*, *Writing, Debugging, and Testing Functions*.

Wrapping error-prone code in a try block

When you know that a statement can cause an error, you should wrap that statement in a `try` block. For example, parsing from text to a number can cause an error. Any statements in the `catch` block will be executed only if an exception is thrown by a statement in the `try` block.

We don't have to do anything inside the `catch` block. Let's see this in action:

1. Use your preferred coding tool to add a new **Console App** / `console` project named `HandlingExceptions` to the `Chapter03` workspace/solution.
    - In Visual Studio Code, select `HandlingExceptions` as the active OmniSharp project.
2. Type statements to prompt the user to enter their age and then write their age to the console, as shown in the following code:

```
WriteLine("Before parsing");
Write("What is your age? ");
string? input = ReadLine(); // or use "49" in a notebook
try
{
  int age = int.Parse(input);
  WriteLine($"You are {age} years old.");
}
catch
{
}
WriteLine("After parsing");
```

You will see the following compiler message:

```
Warning CS8604 Possible null reference argument for parameter 's' in 'int int.Parse(string s)'.
```

By default, in new .NET 6 projects, Microsoft has enabled nullable reference types so you will see many more compiler warnings like this. In production code, you should add code to check for `null` and handle that possibility appropriately. In this book, I will not include these `null` checks because the code samples are not designed to be production-quality and `null` checks everywhere will clutter the code and use up valuable pages.

You will see hundreds of more examples of potentially `null` variables throughout the code samples in this book. Those warnings are safe to ignore for the book code examples. You only need similar warnings when you write your own production code. You will see more about null handling in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*.

In this case, it is impossible for `input` to be `null` because the user must press Enter for `ReadLine` to return and that will return an empty `string`.

1. To disable the compiler warning, change `input` to `input!`. An exclamation mark `!` after an expression is called the **null-forgiving operator** and it disables the compiler warning. But if the expression does evaluate to `null` at runtime, then an exception will be thrown.

This code includes two messages to indicate *before* parsing and *after* parsing to make clearer the flow through the code. These will be especially useful as the example code grows more complex.

1. Run the code, enter `49`, and view the result, as shown in the following output:

```
Before parsing
What is your age? 49
You are 49 years old.
After parsing
```

1. Run the code, enter `Kermit`, and view the result, as shown in the following output:

```
Before parsing
What is your age? Kermit
After parsing
```

When the code was executed, the error exception was caught and the default message and stack trace were not output, and the console app continued running. This is better than the default behavior, but it might be useful to see the type of error that occurred.

> **Good Practice**: You should never use an empty `catch` statement like this in production code because it "swallows" exceptions and hides potential problems. You should at least log the exception if you cannot or do not want to handle it properly, or rethrow it so that higher-level code can decide instead. You will learn about logging in *Chapter 4*, *Writing, Debugging, and Testing Functions*.

Catching all exceptions

To get information about any type of exception that might occur, you can declare a variable of type `System.Exception` to the `catch` block:

1. Add an exception variable declaration to the `catch` block and use it to write information about the exception to the console, as shown in the following code:

```
catch (Exception ex)
{
  WriteLine($"{ex.GetType()} says {ex.Message}");
}
```

1. Run the code, enter `Kermit` again, and view the result, as shown in the following output:

```
Before parsing
What is your age? Kermit
System.FormatException says Input string was not in a correct format.
After parsing
```

Catching specific exceptions

Now that we know which specific type of exception occurred, we can improve our code by catching just that type of exception and customizing the message that we display to the user:

1. Leave the existing `catch` block, and above it, add a new `catch` block for the format exception type, as shown in the following highlighted code:

```
catch (FormatException)
{
  WriteLine("The age you entered is not a valid number format.");
}
catch (Exception ex)
{
  WriteLine($"{ex.GetType()} says {ex.Message}");
}
```

1. Run the code, enter `Kermit` again, and view the result, as shown in the following output:

```
Before parsing
What is your age? Kermit
The age you entered is not a valid number format.
After parsing
```

The reason we want to leave the more general catch below is that there might be other types of exceptions that can occur.

1. Run the code, enter `9876543210`, and view the result, as shown in the following output:

```
Before parsing
What is your age? 9876543210
System.OverflowException says Value was either too large or too small for an Int32.
After parsing
```

Let's add another `catch` block for this type of exception.

1. Leave the existing `catch` blocks, and add a new `catch` block for the overflow exception type, as shown in the following highlighted code:

```
catch (OverflowException)
{
  WriteLine("Your age is a valid number format but it is either too big or small.");
}
catch (FormatException)
{
  WriteLine("The age you entered is not a valid number format.");
}
```

1. Run the code, enter `9876543210`, and view the result, as shown in the following output:

```
Before parsing
What is your age? 9876543210
Your age is a valid number format but it is either too big or small.
After parsing
```

The order in which you catch exceptions is important. The correct order is related to the inheritance hierarchy of the exception types. You will learn about inheritance in *Chapter 5*, *Building Your Own Types with Object-Oriented Programming*. However, don't worry too much about this—the compiler will give you build errors if you get exceptions in the wrong order anyway.

> **Good Practice**: Avoid over-catching exceptions. They should often be allowed to propagate up the call stack to be handled at a level where more information is known about the circumstances that could change the logic of how they should be handled. You will learn about this in *Chapter 4*, *Writing, Debugging, and Testing Functions*.

Catching with filters

You can also add filters to a `catch` statement using the `when` keyword, as shown in the following code:

```
Write("Enter an amount: ");
string amount = ReadLine()!;
if (string.IsNullOrEmpty(amount)) return;
```

```
try
{
  decimal amountValue = decimal.Parse(amount);
  WriteLine($"Amount formatted as currency: {amountValue:C}");
}
catch (FormatException) when (amount.Contains("$"))
{
  WriteLine("Amounts cannot use the dollar sign!");
}
catch (FormatException)
{
  WriteLine("Amounts must only contain digits!");
}
```

Checking for overflow

Earlier, we saw that when casting between number types, it was possible to lose information, for example, when casting from a `long` variable to an `int` variable. If the value stored in a type is too big, it will overflow.

Throwing overflow exceptions with the checked statement

The `checked` statement tells .NET to throw an exception when an overflow happens instead of allowing it to happen silently, which is done by default for performance reasons.

We will set the initial value of an `int` variable to its maximum value minus one. Then, we will increment it several times, outputting its value each time. Once it gets above its maximum value, it overflows to its minimum value and continues incrementing from there. Let's see this in action:

1. Use your preferred coding tool to add a new **Console App** / `console` named `CheckingForOverflow` to the `Chapter03` workspace/solution.
   - In Visual Studio Code, select `CheckingForOverflow` as the active OmniSharp project.
2. In `Program.cs`, delete the existing statements and then type statements to declare and assign an integer to one less than its maximum possible value, and then increment it and write its value to the console three times, as shown in the following code:

```
int x = int.MaxValue - 1;
WriteLine($"Initial value: {x}");
x++;
WriteLine($"After incrementing: {x}");
x++;
WriteLine($"After incrementing: {x}");
x++;
WriteLine($"After incrementing: {x}");
```

1. Run the code and view the result that shows the value overflowing silently and wrapping around to large negative values, as shown in the following output:

```
Initial value: 2147483646
After incrementing: 2147483647
After incrementing: -2147483648
After incrementing: -2147483647
```

1. Now, let's get the compiler to warn us about the overflow by wrapping the statements using a `checked` statement block, as shown highlighted in the following code:

```
checked
{
  int x = int.MaxValue - 1;
  WriteLine($"Initial value: {x}");
  x++;
  WriteLine($"After incrementing: {x}");
  x++;
  WriteLine($"After incrementing: {x}");
  x++;
```

```
  WriteLine($"After incrementing: {x}");
}
```

1. Run the code and view the result that shows the overflow being checked and causing an exception to be thrown, as shown in the following output:

```
Initial value: 2147483646
After incrementing: 2147483647
Unhandled Exception: System.OverflowException: Arithmetic operation resulted in an overflow.
```

1. Just like any other exception, we should wrap these statements in a `try` statement block and display a nicer error message for the user, as shown in the following code:

```
try
{
  // previous code goes here
}
catch (OverflowException)
{
  WriteLine("The code overflowed but I caught the exception.");
}
```

1. Run the code and view the result, as shown in the following output:

```
Initial value: 2147483646
After incrementing: 2147483647
The code overflowed but I caught the exception.
```

Disabling compiler overflow checks with the unchecked statement

The previous section was about the default overflow behavior at *runtime* and how to use the `checked` statement to change that behavior. This section is about *compile time* overflow behavior and how to use the `unchecked` statement to change that behavior.

A related keyword is `unchecked`. This keyword switches off overflow checks performed by the compiler within a block of code. Let's see how to do this:

1. Type the following statement at the end of the previous statements. The compiler will not compile this statement because it knows it would overflow:

```
int y = int.MaxValue + 1;
```

1. Hover your mouse pointer over the error, and note that a compile-time check is shown as an error message, as shown in *Figure 3.1*:

Figure 3.1: A compile-time check in the PROBLEMS window

2. To disable compile-time checks, wrap the statement in an `unchecked` block, write the value of `y` to the console, decrement it, and repeat, as shown in the following code:

```
unchecked
{
  int y = int.MaxValue + 1;
  WriteLine($"Initial value: {y}");
  y--;
  WriteLine($"After decrementing: {y}");
  y--;
  WriteLine($"After decrementing: {y}");
}
```

1. Run the code and view the results, as shown in the following output:

```
Initial value: -2147483648
After decrementing: 2147483647
After decrementing: 2147483646
```

Of course, it would be rare that you would want to explicitly switch off a check like this because it allows an overflow to occur. But perhaps you can think of a scenario where you might want that behavior.

Practicing and exploring

Test your knowledge and understanding by answering some questions, getting some hands-on practice, and exploring with deeper research into this chapter's topics.

Exercise 3.1 – Test your knowledge

Answer the following questions:

1. What happens when you divide an `int` variable by `0`?
2. What happens when you divide a `double` variable by `0`?
3. What happens when you overflow an `int` variable, that is, set it to a value beyond its range?
4. What is the difference between `x = y++;` and `x = ++y;`?
5. What is the difference between `break`, `continue`, and `return` when used inside a loop statement?
6. What are the three parts of a `for` statement and which of them are required?
7. What is the difference between the `=` and `==` operators?
8. Does the following statement compile?

```
for ( ; ; ) ;
```

1. What does the underscore `_` represent in a `switch` expression?
2. What interface must an object "implement" to be enumerated over by using the `foreach` statement?

Exercise 3.2 – Explore loops and overflow

What will happen if this code executes?

```
int max = 500;
for (byte i = 0; i < max; i++)
{
  WriteLine(i);
}
```

Create a console app in `Chapter03` named `Ch03Ex02LoopsAndOverflow` and enter the preceding code. Run the console app and view the output. What happens?

What code could you add (don't change any of the preceding code) to warn us about the problem?

Exercise 3.3 – Practice loops and operators

FizzBuzz is a group game for children to teach them about division. Players take turns to count incrementally, replacing any number divisible by three with the word *fizz*, any number divisible by five with the word *buzz*, and any number divisible by both with *fizzbuzz*.

Create a console app in `Chapter03` named `Ch03Ex03FizzBuzz` that outputs a simulated FizzBuzz game that counts up to 100. The output should look something like *Figure 3.2*:

Figure 3.2: A simulated FizzBuzz game output

Exercise 3.4 – Practice exception handling

Create a console app in `Chapter03` named `Ch03Ex04Exceptions` that asks the user for two numbers in the range 0-255 and then divides the first number by the second:

```
Enter a number between 0 and 255: 100
Enter another number between 0 and 255: 8
100 divided by 8 is 12
```

Write exception handlers to catch any thrown errors, as shown in the following output:

```
Enter a number between 0 and 255: apples
Enter another number between 0 and 255: bananas
FormatException: Input string was not in a correct format.
```

Exercise 3.5 – Test your knowledge of operators

What are the values of `x` and `y` after the following statements execute? Create a console app in `Chapter03` named `Ch03Ex05Operators` to test your assumptions.

1. Increment and addition operators:

```
x = 3;
y = 2 + ++x;
```

1. Binary shift operators:

```
x = 3 << 2;
y = 10 >> 1;
```

1. Bitwise operators:

```
x = 10 & 8;
y = 10 | 7;
```

Exercise 3.6 – Explore topics

Use the links on the following page to learn about the topics covered in this chapter in more detail:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-3---controlling-flow-converting-types-and-handling-exceptions

Summary

In this chapter, you learned how to:

- Use operators to perform simple tasks.
- Use branch and loop statements to implement logic.
- Work with single- and multi-dimensional arrays.
- Convert between types.
- Catch exceptions.

You are now ready to learn how to reuse blocks of code by defining functions, how to pass values into them and get values back, and how to track down bugs in your code and squash them using debugging and testing tools!

4 Writing, Debugging, and Testing Functions

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about writing functions to reuse code, debugging logic errors during development, logging exceptions during runtime, unit testing your code to remove bugs, and ensuring stability and reliability.

This chapter covers the following topics:

- Writing functions
- Debugging during development
- Hot reloading during development
- Logging during runtime
- Unit testing
- Throwing and catching exceptions in functions

Writing functions

A fundamental principle of programming is **Don't Repeat Yourself** (**DRY**).

While programming, if you find yourself writing the same statements over and over again, then turn those statements into a function. Functions are like tiny programs that complete one small task. For example, you might write a function to calculate sales tax and then reuse that function in many places in a financial application.

Like programs, functions usually have inputs and outputs. They are sometimes described as black boxes, where you feed some raw materials in one end, and a manufactured item emerges at the other. Once created and thoroughly debugged and tested, you don't need to think about how they work.

Understanding top-level programs and functions

In *Chapter 1*, *Hello, C#! Welcome, .NET!*, we learned that since .NET 6, the default project template for console apps uses the top-level program feature introduced with C# 9.

Once you start writing functions, it is important to understand how they work with the automatically generated `Program` class and its `<Main>$` method.

In `Program.cs`, you might write statements to import a class, call one of its methods, and define and call a function, as shown in the following code:

```
using static System.Console;
WriteLine("Hello, World!");
DoSomething(); // call the function
void DoSomething() // define a function
{
  WriteLine("Doing something!");
}
```

The compiler automatically generates a `Program` class with a `<Main>$` function, and then moves your statements and function inside the `<Main>$` method, and renames the function, as shown highlighted in the following code:

```
using static System.Console;
partial class Program
{
  static void <Main>$(String[] args)
  {
    WriteLine("Hello, World!");
    <<Main>$>g__DoSomething|0_0(); // call the function
    void <<Main>$>g__DoSomething|0_0() // define a local function
    {
      WriteLine("Doing something!");
    }
  }
}
```

For the compiler to know what statements need to go where, you must follow some rules:

- Import statements (`using`) must go at the top of the `Program.cs` file.
- Statements that will go in the `<Main>$` function must go in the middle of the `Program.cs` file.
- Functions must go at the bottom of the `Program.cs` file. They will become **local functions**.

The last point is important because local functions have limitations, as you will see later in this chapter.

You are about to see some C# keywords like `static` and `partial` that will be formally introduced in *Chapter 5*, *Building Your Own Types with Object-Oriented Programming*.

A better approach is to define the function in a separate file and to add it as a `static` member of the `Program` class, as shown in the following code:

```
// in a file named Program.Functions.cs
partial class Program
{
  static void DoSomething() // define a non-local static function
  {
    WriteLine("Doing something!");
  }
}
// in the Program.cs file
using static System.Console;
WriteLine("Hello, World!");
DoSomething(); // call the function
```

The compiler defines a `Program` class with a `<Main>$` function and moves your statements inside the `<Main>$` method, and merges your function as a member of the `Program` class, as shown in the following highlighted code:

```
using static System.Console;
partial class Program
{
  static void <Main>$(String[] args)
  {
    WriteLine("Hello, World!");
    DoSomething(); // call the function
  }
  static void DoSomething() // define a function
  {
    WriteLine("Doing something!");
  }
}
```

Good Practice: Create any functions that you will call in `Program.cs` in a separate file and manually define them inside a `partial Program` class. This will merge them into the automatically generated `Program` class *at the same level* as the `<Main>$` method, instead of as local functions *inside* the `<Main>$` method.

Times table example

Let's say that you want to help your child learn their times tables, so you want to make it easy to generate a times table for a number, such as the 7 times table:

```
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
...
10 x 7 = 70
11 x 7 = 77
12 x 7 = 84
```

Most times tables have either 10, 12, or 20 rows, depending on how advanced the child is.

You learned about the `for` statement earlier in this book, so you know that it can be used to generate repeated lines of output when there is a regular pattern, such as a 7 times table with 12 rows, as shown in the following code:

```
for (int row = 1; row <= 12; row++)
{
  Console.WriteLine($"{row} x 7 = {row * 7}");
}
```

However, instead of outputting the 7 times table always with 12 rows, we want to make this more flexible so it can output any size times table for any number. We can do this by creating a function.

Writing a times table function

Let's explore functions by creating one to output any times table for numbers 0 to 255 of any size up to 255 rows (but it defaults to 12 rows):

1. Use your preferred coding tool to create a new project, as defined in the following list:
   - Project template: **Console App** / `console`
   - Project file and folder: `WritingFunctions`
   - Workspace/solution file and folder: `Chapter04`
2. Open `WritingFunctions.csproj`, and after the `<PropertyGroup>` section, add a new `<ItemGroup>` section to statically import `System.Console` for all C# files using the implicit usings .NET SDK feature, as shown in the following markup:

```
<ItemGroup>
  <Using Include="System.Console" Static="true" />
</ItemGroup>
```

1. Add a new class file to the project named `Program.Functions.cs`.
   - In Visual Studio 2022, navigate to **Project** | **Add Class…**, type the name and then click **Add**.
   - In Visual Studio Code, click the **New File…** button.
2. In `Program.Functions.cs`, write statements to define a function named `TimesTable` in the partial `Program` class, as shown in the following code:

```
partial class Program
{
  static void TimesTable(byte number, byte size = 12)
  {
    WriteLine($"This is the {number} times table with {size} rows:");
    for (int row = 1; row <= size; row++)
    {
      WriteLine($"{row} x {number} = {row * number}");
    }
    WriteLine();
  }
}
```

In the preceding code, note the following:

- `TimesTable` must have a `byte` value passed to it as a parameter named `number`.
- `TimesTable` can optionally have a `byte` value passed to it as a parameter named `size`. If a value is not passed, it defaults to `12`.
- `TimesTable` is a `static` method because it will be called by the `static` method `<Main>$`.
- `TimesTable` does not return a value to the caller, so it is declared with the `void` keyword before its name.
- `TimesTable` uses a `for` statement to output the times table for the `number` passed to it with a number of rows equal to `size`.

1. In `Program.cs`, delete the existing statements and then call the function. Pass in a `byte` value for the `number` parameter, for example, `7`, as shown in the following code:

```
TimesTable(7);
```

1. Run the code and then view the result, as shown in the following output:

```
This is the 7 times table with 12 rows:
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
6 x 7 = 42
7 x 7 = 49
8 x 7 = 56
9 x 7 = 63
10 x 7 = 70
11 x 7 = 77
12 x 7 = 84
```

1. Set the `size` parameter to `20`, as shown in the following code:

```
TimesTable(7, 20);
```

1. Run the console app and confirm that the times table now has twenty rows.

   Good Practice: If a function has one or more parameters where just passing the values may not provide enough meaning, then you can optionally specify the name of the parameter as well as its value, as shown in the following code: `TimesTable(number: 7, size: 10)`.

2. Change the number passed into the `TimesTable` function to other `byte` values between `0` and `255` and confirm that the output times tables are correct.
3. Note that if you try to pass a non-`byte` number, for example, an `int` or `double` or `string`, an error is returned, as shown in the following output:

```
Error: (1,12): error CS1503: Argument 1: cannot convert from 'int' to 'byte'
```

Writing a function that returns a value

The previous function performed actions (looping and writing to the console), but it did not return a value. Let's say that you need to calculate sales or value-added tax (VAT). In Europe, VAT rates can range from 8% in Switzerland to 27% in Hungary. In the United States, state sales taxes can range from 0% in Oregon to 8.25% in California.

Tax rates change all the time, and they vary based on many factors. The values used in this example do not need to be accurate.

Let's implement a function to calculate taxes in various regions around the world:

1. In `Program.Functions.cs`, write a function named `CalculateTax`, as shown in the following code:

```
partial class Program
{
  ...
  static decimal CalculateTax(
    decimal amount, string twoLetterRegionCode)
  {
    decimal rate = 0.0M;
    switch (twoLetterRegionCode)
    {
      case "CH": // Switzerland
        rate = 0.08M;
        break;
      case "DK": // Denmark
      case "NO": // Norway
        rate = 0.25M;
        break;
      case "GB": // United Kingdom
      case "FR": // France
        rate = 0.2M;
        break;
      case "HU": // Hungary
        rate = 0.27M;
        break;
      case "OR": // Oregon
      case "AK": // Alaska
      case "MT": // Montana
        rate = 0.0M;
        break;
      case "ND": // North Dakota
      case "WI": // Wisconsin
      case "ME": // Maine
      case "VA": // Virginia
        rate = 0.05M;
        break;
      case "CA": // California
        rate = 0.0825M;
        break;
      default: // most US states
        rate = 0.06M;
        break;
    }
    return amount * rate;
  }
}
```

In the preceding code, note the following:

- `CalculateTax` has two inputs: a parameter named `amount` that will be the amount of money spent, and a parameter named `twoLetterRegionCode` that will be the region the amount is spent in.
- `CalculateTax` will perform a calculation using a `switch` statement and then return the sales tax or VAT owed on the amount as a `decimal` value; so, before the name of the function, we have declared the data type of the return value to be `decimal`.

1. Comment out any `TimesTable` method calls and then call the `CalculateTax` method, passing values for the amount such as `149` and a valid region code such as `FR`, as shown in the following code:

```
decimal taxToPay = CalculateTax(amount: 149, twoLetterRegionCode: "FR");
WriteLine($"You must pay {taxToPay} in tax.");
```

1. Run the code and view the result, as shown in the following output:

```
You must pay 29.8 in tax.
```

We could format the `taxToPay` output as currency by using `{taxToPay:C}`, but it will use your local culture to decide how to format the currency symbol and decimals. For example, for me in the UK, I would see `£29.80`.

Can you think of any problems with the `CalculateTax` function as written? What would happen if the user enters a code such as `fr` or `UK`? How could you rewrite the function to improve it? Would using a `switch` *expression* instead of a `switch` *statement* be clearer?

Converting numbers from cardinal to ordinal

Numbers that are used to count are called **cardinal** numbers, for example, 1, 2, and 3, whereas numbers used to order are **ordinal** numbers, for example, 1st, 2nd, and 3rd. Let's create a function to convert cardinals to ordinals:

1. In `Program.Functions.cs`, write a function named `CardinalToOrdinal` that converts a cardinal `int` value into an ordinal `string` value; for example, it converts 1 into 1st, 2 into 2nd, and so on, as shown in the following code:

```
static string CardinalToOrdinal(int number)
{
  switch (number)
  {
    case 11: // special cases for 11th to 13th
    case 12:
    case 13:
      return $"{number}th";
    default:
      int lastDigit = number % 10;
      string suffix = lastDigit switch
      {
        1 => "st",
        2 => "nd",
        3 => "rd",
        _ => "th"
      };
      return $"{number}{suffix}";
  }
}
```

From the preceding code, note the following:

- `CardinalToOrdinal` has one input: a parameter of the `int` type named `number`, and one output: a return value of the `string` type.
- A `switch` *statement* is used to handle the special cases of 11, 12, and 13.
- A `switch` *expression* then handles all other cases: if the last digit is 1, then use `st` as the suffix; if the last digit is 2, then use `nd` as the suffix; if the last digit is 3, then use `rd` as the suffix; and if the last digit is anything else, then use `th` as the suffix.

1. In `Program.Functions.cs`, write a function named `RunCardinalToOrdinal` that uses a `for` statement to loop from 1 to 40, calling the `CardinalToOrdinal` function for each number and writing the returned `string` to the console, separated by a space character, as shown in the following code:

```
static void RunCardinalToOrdinal()
{
  for (int number = 1; number <= 40; number++)
  {
    Write($"{CardinalToOrdinal(number)} ");
  }
  WriteLine();
}
```

1. In `Program.cs`, comment out the `CalculateTax` statements, and call the `RunCardinalToOrdinal` method, as shown in the following code:

```
RunCardinalToOrdinal();
```

1. Run the console app and view the results, as shown in the following output:

```
1st 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th 12th 13th 14th 15th 16th 17th 18th 19th 20th 21st 2
```

Calculating factorials with recursion

The factorial of 5 is 120, because factorials are calculated by multiplying the starting number by one less than itself, and then by one less again, and so on, until the number is reduced to 1. An example can be seen here: 5 x 4 x 3 x 2 x 1 = 120.

The factorial function is defined for non-negative integers only, i.e. for 0, 1, 2, 3, and so on, and it is defined as:

```
0! = 1
n! = n × (n − 1)!, for n ∈ { 1, 2, 3, ... }
```

Factorials are written like this: 5!, where the exclamation mark is read as "bang," so 5! = 120, that is, *five bang equals one hundred and twenty*. Bang is a good name for factorials because they increase in size very rapidly, just like an explosion.

We will write a function named `Factorial`; this will calculate the factorial for an `int` passed to it as a parameter. We will use a clever technique called **recursion**, which means a function that calls itself within its implementation, either directly or indirectly:

1. In `Program.Functions.cs`, write a function named `Factorial`, as shown in the following code:

```
static int Factorial(int number)
{
  if (number < 0)
  {
     throw new ArgumentException(message:
       "The factorial function is defined for non-negative integers only.",
       paramName: "number");
  }
  else if (number == 0)
  {
    return 1;
  }
  else
  {
    return number * Factorial(number - 1);
  }
}
```

As before, there are several noteworthy elements of the preceding code, including the following:

- If the input parameter `number` is negative, `Factorial` throws an exception.
- If the input parameter `number` is zero, `Factorial` returns `1`.
- If the input parameter `number` is more than `1`, which it will be in all other cases, `Factorial` multiplies the number by the result of calling itself and passing one less than `number`. This makes the function recursive.

More Information: Recursion is clever, but it can lead to problems, such as a stack overflow due to too many function calls because memory is used to store data on every function call, and it eventually uses too much. Iteration is a more practical, if less succinct, solution in languages such as C#. You can read more about this at the following link: https://en.wikipedia.org/wiki/Recursion_(computer_science)#Recursion_versus_iteration.

1. In `Program.Functions.cs`, write a function named `RunFactorial` that uses a `for` statement to output the factorials of numbers from 1 to 14, calls the `Factorial` function inside its loop, and then outputs the result, formatted using the code `N0`, which means number format using thousand separators with zero decimal places, as shown in the following code:

```
static void RunFactorial()
{
  for (int i = 1; i <= 14; i++)
  {
    WriteLine($"{i}! = {Factorial(i):N0}");
```

```
  }
}
```

1. Comment out the `RunCardinalToOrdinal` method call and call the `RunFactorial` method.
2. Run the code and view the results, as shown in the following output:

```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5,040
8! = 40,320
9! = 362,880
10! = 3,628,800
11! = 39,916,800
12! = 479,001,600
13! = 1,932,053,504
14! = 1,278,945,280
```

It is not immediately obvious in the previous output, but factorials of 13 and higher overflow the `int` type because they are so big. 12! is 479,001,600, which is about half a billion. The maximum positive value that can be stored in an `int` variable is about two billion. 13! is 6,227,020,800, which is about six billion, and when stored in a 32-bit integer it overflows silently without showing any problems.

What should you do to get notified when an overflow happens? Of course, we could solve the problem for 13! and 14! by using a `long` (64-bit integer) instead of an `int` (32-bit integer), but we will quickly hit the overflow limit again.

The point of this section is to understand that numbers can overflow and how to show that rather than ignore it, not specifically how to calculate factorials higher than 12!.

1. Modify the `Factorial` function to check for overflows, as shown highlighted in the following code:

```
checked // for overflow
{
  return number * Factorial(number - 1);
}
```

1. Modify the `RunFactorial` function to handle overflow exceptions when calling the `Factorial` function, as shown highlighted in the following code:

```
try
{
  WriteLine($"{i}! = {Factorial(i):N0}");
}
catch (System.OverflowException)
{
  WriteLine($"{i}! is too big for a 32-bit integer.");
}
```

1. Run the code and view the results, as shown in the following output:

```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5,040
8! = 40,320
9! = 362,880
10! = 3,628,800
11! = 39,916,800
12! = 479,001,600
```

```
13! is too big for a 32-bit integer.
14! is too big for a 32-bit integer.
```

Documenting functions with XML comments

By default, when calling a function such as `CardinalToOrdinal`, code editors will show a tooltip with basic information, as shown in *Figure 4.1*:

Figure 4.1: A tooltip showing the default simple method signature

Let's improve the tooltip by adding extra information:

1. If you are using Visual Studio Code with the **C#** extension, you should navigate to **View | Command Palette | Preferences: Open Settings (UI)**, and then search for `formatOnType` and make sure that is enabled. C# XML documentation comments are a built-in feature of Visual Studio 2022.
2. On the line above the `CardinalToOrdinal` function, type three forward slashes `///`, and note that they are expanded into an XML comment that recognizes that the function has a single parameter named `number`, as shown in the following code:

```
/// <summary>
///
/// </summary>
/// <param name="number"></param>
/// <returns></returns>
```

1. Enter suitable information for the XML documentation comment for the `CardinalToOrdinal` function. Add a summary, and describe the input parameter and the return value, as shown in the following code:

```
/// <summary>
/// Pass a 32-bit integer and it will be converted into its ordinal equivalent.
```

```
/// </summary>
/// <param name="number">Number as a cardinal value e.g. 1, 2, 3, and so on.</param>
/// <returns>Number as an ordinal value e.g. 1st, 2nd, 3rd, and so on.</returns>
```

1. Now, when calling the function, you will see more details, as shown in *Figure 4.2*:

Figure 4.2: A tooltip showing the more detailed method signature

It is worth emphasizing that this feature is primarily designed to be used with a tool that converts the comments into documentation like Sandcastle, which you can read more about at the following link: https://github.com/EWSoftware/SHFB. The tooltips that appear while entering code or hovering over the function name are a secondary feature.

Local functions do not support XML comments because local functions cannot be used outside the member in which they are declared, so it makes no sense to generate documentation from them. Sadly, this also means no tooltip, which would still be useful, but neither Visual Studio 2022 nor Visual Studio Code recognize that.

Good Practice: Add XML documentation comments to all your functions except local functions.

Using lambdas in function implementations

F# is Microsoft's strongly typed functional-first programming language that, like C#, compiles to IL to be executed by .NET. Functional languages evolved from lambda calculus, a computational system based only on functions. The code looks more like mathematical functions than steps in a recipe.

Some of the important attributes of functional languages are defined in the following list:

- **Modularity**: The same benefit of defining functions in C# applies to functional languages. Break up a large complex code base into smaller pieces.
- **Immutability**: Variables in the C# sense do not exist. Any data value inside a function cannot change. Instead, a new data value can be created from an existing one. This reduces bugs.
- **Maintainability**: Code is cleaner and clearer (for mathematically inclined programmers!).

Since C# 6, Microsoft has worked to add features to the language to support a more functional approach. For example, adding **tuples** and **pattern matching** in C# 7, **non-null reference types** in C# 8, and improving pattern matching and adding records, that is, **immutable objects**, in C# 9.

In C# 6, Microsoft added support for **expression-bodied function members**. We will look at an example of this now. In C#, lambdas are the use of the `=>` character to indicate a return value from a function.

The **Fibonacci sequence** of numbers always starts with 0 and 1. Then the rest of the sequence is generated using the rule of adding together the previous two numbers, as shown in the following sequence of numbers:

```
0 1 1 2 3 5 8 13 21 34 55 ...
```

The next term in the sequence would be 34 + 55, which is 89.

We will use the Fibonacci sequence to illustrate the difference between an imperative and declarative function implementation:

1. In `Program.Functions.cs`, write a function named `FibImperative` that will be written in an imperative style, as shown in the following code:

```
static int FibImperative(int term)
{
  if (term == 1)
  {
    return 0;
  }
  else if (term == 2)
  {
    return 1;
  }
  else
  {
    return FibImperative(term - 1) + FibImperative(term - 2);
  }
}
```

1. In `Program.Functions.cs`, write a function named `RunFibImperative` that calls `FibImperative` inside a `for` statement that loops from 1 to 30, as shown in the following code:

```
static void RunFibImperative()
{
  for (int i = 1; i <= 30; i++)
  {
    WriteLine("The {0} term of the Fibonacci sequence is {1:N0}.",
      arg0: CardinalToOrdinal(i),
      arg1: FibImperative(term: i));
  }
}
```

1. In `Program.cs`, comment out the other method calls and call the `RunFibImperative` method.
2. Run the console app and view the results, as shown in the following partial output:

```
The 1st term of the Fibonacci sequence is 0.
The 2nd term of the Fibonacci sequence is 1.
The 3rd term of the Fibonacci sequence is 1.
The 4th term of the Fibonacci sequence is 2.
The 5th term of the Fibonacci sequence is 3.
The 6th term of the Fibonacci sequence is 5.
```

```
The 7th term of the Fibonacci sequence is 8.
...
The 28th term of the Fibonacci sequence is 196,418.
The 29th term of the Fibonacci sequence is 317,811.
The 30th term of the Fibonacci sequence is 514,229.
```

1. In `Program.Functions.cs`, write a function named `FibFunctional` written in a declarative style, as shown in the following code:

```
static int FibFunctional(int term) =>
  term switch
  {
    1 => 0,
    2 => 1,
    _ => FibFunctional(term - 1) + FibFunctional(term - 2)
  };
```

1. In `Program.Functions.cs`, write a function to call it inside a `for` statement that loops from 1 to 30, as shown in the following code:

```
static void RunFibFunctional()
{
  for (int i = 1; i <= 30; i++)
  {
    WriteLine("The {0} term of the Fibonacci sequence is {1:N0}.",
      arg0: CardinalToOrdinal(i),
      arg1: FibFunctional(term: i));
  }
}
```

1. In `Program.cs`, comment out the `RunFibImperative` method call, and call the `RunFibFunctional` method.
2. Run the code and view the results (which will be the same as before).

Debugging during development

In this section, you will learn how to debug problems at development time. You must use a code editor that has debugging tools such as Visual Studio 2022 or Visual Studio Code. At the time of writing, you cannot use .NET Interactive Notebooks to debug code, but this is expected to be added in the future.

> **More Information**: Some people find it tricky setting up the OmniSharp debugger for Visual Studio Code. I have included instructions for the most common issues, but if you still have trouble, try reading the information at the following link: https://github.com/OmniSharp/omnisharp-vscode/blob/master/debugger.md

Using Visual Studio Code integrated terminal during debugging

By default, OmniSharp sets the console to use the internal console during debugging, which does not allow interactions like entering text from the `ReadLine` method.

To improve the experience, we can change a setting to use the integrated terminal instead:

1. In any project that you want to set breakpoints and step through code, in the `.vscode` folder, open the `launch.json` file.
2. Change the `console` setting from `internalConsole` to `integratedTerminal`, as shown highlighted in the following partial configuration:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      ...
      "console": "integratedTerminal",
      ...
```

```
  ]
}
```

Creating code with a deliberate bug

Let's explore debugging by creating a console app with a deliberate bug that we will then use the debugger tools in your code editor to track down and fix:

1. Use your preferred coding tool to add a new **Console App** / `console` project named `Debugging` to the `Chapter04` workspace/solution.
   - In Visual Studio Code, select `Debugging` as the active OmniSharp project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.
   - In Visual Studio 2022, set the startup project for the solution to the current selection.
2. Modify `Debugging.csproj` to statically import `System.Console` for all code files.
3. In `Program.cs`, at the bottom of the file, add a function with a deliberate bug, as shown in the following code:

```
double Add(double a, double b)
{
  return a * b; // deliberate bug!
}
```

1. Above the `Add` function, write statements to declare and set some variables and then add them together using the buggy function, as shown in the following code:

```
double a = 4.5;
double b = 2.5;
double answer = Add(a, b);
WriteLine($"{a} + {b} = {answer}");
WriteLine("Press ENTER to end the app.");
ReadLine(); // wait for user to press ENTER
```

1. Run the console application and view the result, as shown in the following output:

```
4.5 + 2.5 = 11.25
Press ENTER to end the app.
```

But wait, there's a bug! 4.5 added to 2.5 should be 7, not 11.25!

We will use the debugging tools to hunt for and squish the bug.

Setting a breakpoint and start debugging

Breakpoints allow us to mark a line of code that we want to pause at to inspect the program state and find bugs.

Using Visual Studio 2022

Let's set a breakpoint and then start debugging using Visual Studio 2022:

1. Click in line 1, which is the statement that declares the variable named `a`.
2. Navigate to **Debug** | **Toggle Breakpoint** or press F9. A red circle will appear in the margin bar on the left-hand side and the statement will be highlighted in red to indicate that a breakpoint has been set, as shown in *Figure 4.3*:

Figure 4.3: Toggling breakpoints using Visual Studio 2022

Breakpoints can be toggled off with the same action. You can also left-click in the margin to toggle a breakpoint on and off, or right-click a breakpoint to see more options, such as delete, disable, or edit conditions or actions for an existing breakpoint.

1. Navigate to **Debug | Start Debugging** or press F5. Visual Studio starts the console application and then pauses when it hits the breakpoint. This is known as break mode. Extra windows titled **Locals** (showing current values of local variables), **Watch 1** (showing any watch expressions you have defined), **Call Stack**, **Exception Settings**, and **Immediate Window** appear. The **Debugging** toolbar appears. The line that will be executed next is highlighted in yellow, and a yellow arrow points at the line from the margin bar, as shown in *Figure 4.4*:

Figure 4.4: Break mode in Visual Studio 2022

If you do not want to see how to use Visual Studio Code to start debugging, then you can skip the next section and continue to the section titled *Navigating with the debugging toolbar*.

Using Visual Studio Code

Let's set a breakpoint and then start debugging using Visual Studio Code:

1. Click in line 1, which is the statement that declares the variable named `a`.
2. Navigate to **Run** | **Toggle Breakpoint** or press F9. A red circle will appear in the margin bar on the left-hand side to indicate that a breakpoint has been set.

Breakpoints can be toggled off with the same action. You can also left-click in the margin to toggle a breakpoint on and off, or right-click to see more options, such as remove, edit, or disable an existing breakpoint; or add a breakpoint, conditional breakpoint, or logpoint when a breakpoint does not yet exist.

Logpoints, also known as **tracepoints**, indicate that you want to record some information without having to stop executing the code at that point.

1. Navigate to **View** | **Run**, or in the left navigation bar you can click the **Run and Debug** icon (the triangle "play" button and "bug").

2. At the top of the **RUN AND DEBUG** window, click on the dropdown to the right of the **Start Debugging** button (green triangular "play" button), and select **.NET Core Launch (console) (Debugging)**, as shown in *Figure 4.5*:

Figure 4.5: Selecting the project to debug using Visual Studio Code

Good Practice: If you do not see a choice in the dropdown list for the **Debugging** project, it is because that project does not have the assets needed to debug. Those assets are stored in the `.vscode` folder. To create the `.vscode` folder for a project, navigate to **View** | **Command Palette**, select **OmniSharp: Select Project**, and then select the **Debugging** project. After a few seconds, when prompted, **Required assets to build and debug are missing from 'Debugging'. Add them?**, click **Yes** to add the missing assets.

3. At the top of the **RUN AND DEBUG** window, click the **Start Debugging** button (green triangular "play" button), or navigate to **Run** | **Start Debugging**, or press F5. Visual Studio Code starts the console app and then pauses when it hits the breakpoint. This is known as **break mode**. The line that will be executed next is highlighted in yellow, and a yellow block points at the line from the margin bar, as shown in *Figure 4.6*:

Figure 4.6: Break mode in Visual Studio Code

Navigating with the debugging toolbar

Visual Studio Code shows a floating toolbar with buttons to make it easy to access debugging features. Visual Studio 2022 has one button in its **Standard** toolbar to start or continue debugging and a separate **Debugging** toolbar for the rest of the tools.

Both are shown in *Figure 4.7* and as described in the following list:

Figure 4.7: Debugging toolbars in Visual Studio 2022 and Visual Studio Code

- **Continue**/F5: This button will continue running the program from the current position until it ends or hits another breakpoint.
- **Step Over**/F10, **Step Into**/F11, and **Step Out**/Shift + F11 (blue arrows over dots): These buttons step through the code statements in various ways, as you will see in a moment.

- **Restart**/Ctrl or Cmd + Shift + F5 (circular arrow): This button will stop and then immediately restart the program with the debugger attached again.
- **Stop**/Shift + F5 (red square): This button will stop the debugging session.

Debugging windows

While debugging, both Visual Studio Code and Visual Studio 2022 show extra windows that allow you to monitor useful information, such as variables, while you step through your code.

The most useful windows are described in the following list:

- **VARIABLES**, including **Locals**, which shows the name, value, and type for any local variables automatically. Keep an eye on this window while you step through your code.
- **WATCH**, or **Watch 1**, which shows the value of variables and expressions that you manually enter.
- **CALL STACK**, which shows the stack of function calls.
- **BREAKPOINTS**, which shows all your breakpoints and allows finer control over them.

When in break mode, there is also a useful window at the bottom of the edit area:

DEBUG CONSOLE or **Immediate Window** enables live interaction with your code. You can interrogate the program state, for example, by entering the name of a variable. For example, you can ask a question such as, "What is 1+2?" by typing `1+2` and pressing Enter.

Stepping through code

Let's explore some ways to step through the code using either Visual Studio or Visual Studio Code:

1. Navigate to **Run** or **Debug** | **Step Into**, or click on the **Step Into** button in the toolbar, or press F11. The yellow highlight steps forward one line.
2. Navigate to **Run** or **Debug** | **Step Over**, or click on the **Step Over** button in the toolbar, or press F10. The yellow highlight steps forward one line. At the moment, you can see that there is no difference between using **Step Into** or **Step Over**.
3. You should now be on the line that calls the `Add` method.

The difference between **Step Into** and **Step Over** can be seen when you are about to execute a method call:

- If you click on **Step Into**, the debugger steps *into* the method so that you can step through every line in that method.
- If you click on **Step Over**, the whole method is executed in one go; it does not skip over the method without executing it.

1. Click on **Step Into** to step inside the method.
2. Hover your mouse pointer over the `a` or `b` parameters in the code editing window and note that a tooltip appears showing their current value.
3. Select the expression `a * b`, right-click the expression, and select **Add to Watch** or **Add Watch**. The expression is added to the **WATCH** or **Watch 1** window, showing that this operator is multiplying `a` by `b` to give the result `11.25`.
4. In the **WATCH** or **Watch 1** window, right-click the expression and choose **Remove Expression** or **Delete Watch**.
5. Fix the bug by changing `*` to `+` in the `Add` function.
6. Stop debugging, recompile, and restart debugging by clicking the circular arrow **Restart** button or pressing Ctrl or Cmd + Shift + F5.
7. Step over the function, take a minute to note how it now calculates correctly, and click the **Continue** button or press F5.
8. With Visual Studio Code, note that when writing to the console during debugging, the output appears in the **DEBUG CONSOLE** window instead of the **TERMINAL** window, as shown in *Figure 4.8*:

Figure 4.8: Writing to the DEBUG CONSOLE during debugging

Customizing breakpoints

It is easy to make more complex breakpoints:

1. If you are still debugging, click the **Stop** button in the debugging toolbar, or navigate to **Run** or **Debug** | **Stop Debugging**, or press Shift + F5.
2. Navigate to **Run** | **Remove All Breakpoints** or **Debug** | **Delete All Breakpoints**.
3. Click on the `WriteLine` statement that outputs the answer.
4. Set a breakpoint by pressing F9 or navigating to **Run** or **Debug** | **Toggle Breakpoint**.
5. Right-click the breakpoint and choose the appropriate menu for your code editor:
    - In Visual Studio Code, choose **Edit Breakpoint...**
    - In Visual Studio 2022, choose **Conditions...**.

6. Type an expression, such as the `answer` variable must be greater than 9, and then press Enter to accept it, and note the expression must evaluate to true for the breakpoint to activate, as shown in *Figure 4.9*:

Figure 4.9: Customizing a breakpoint with an expression using Visual Studio Code

7. Start debugging and note the breakpoint is not hit.
8. Stop debugging.
9. Edit the breakpoint or its conditions and change its expression to less than 9.
10. Start debugging and note the breakpoint is hit.
11. Stop debugging.

12. Edit the breakpoint or its conditions (in Visual Studio 2022 click **Add condition**), and select **Hit Count**, then enter a number such as `3`, meaning that you would have to hit the breakpoint three times before it activates, as shown in *Figure 4.10*:

Figure 4.10: Customizing a breakpoint with an expression and hot count using Visual Studio 2022

13. Hover your mouse over the breakpoint's red circle to see a summary, as shown in *Figure 4.11*:

Figure 4.11: A summary of a customized breakpoint in Visual Studio Code

You have now fixed a bug using some debugging tools and seen some advanced possibilities for setting breakpoints.

Hot reloading during development

Hot Reload is a feature that allows a developer to apply changes to code while the app is running and immediately see the effect. This is great for fixing bugs quickly. Hot Reload is also known as **Edit and Continue**. A list of the types of changes that you can make that support Hot Reload are found at the following link: https://aka.ms/dotnet/hot-reload

Just before the release of .NET 6, a high-level Microsoft employee caused controversy by attempting to make the feature Visual Studio-only. Luckily the open source contingent within Microsoft successfully had the decision overturned. Hot Reload remains available using the command line tool as well.

Let's see it in action:

1. Use your preferred coding tool to add a new **Console App** / `console` project named `HotReloading` to the `Chapter04` workspace/solution.
   - In Visual Studio Code, select `HotReloading` as the active OmniSharp project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.
2. Modify `HotReloading.csproj` to statically import `System.Console` for all code files.
3. In `Program.cs`, delete the existing statements and then write a message to console output every two seconds, as shown in the following code:

```
/* Visual Studio: run the app, change the message, click Hot Reload button.
 * Visual Studio Code: run the app using dotnet watch, change the message. */
while (true)
{
  WriteLine("Hello, Hot Reload!");
  await Task.Delay(2000);
}
```

Hot reloading using Visual Studio 2022

If you are using Visual Studio, working with Hot Reload is built-in to the user interface:

1. In Visual Studio, start the console app, and note that the message is output every two seconds.
2. Change `Hello` to `Goodbye`, navigate to **Debug** | **Apply Code Changes** or click the **Hot Reload** button in the toolbar, and note the change is applied without needing to restart the console app.

3. Drop down the **Hot Reload** button menu and select **Hot Reload on File Save**, as shown in *Figure 4.*:

Figure 4.12: Changing Hot Reload options

4. Change the message again, save the file, and note the console app updates automatically.

Hot reloading using Visual Studio Code and the command line

If you are using Visual Studio Code, you must issue a special command when starting the console app to activate Hot Reload:

1. In Visual Studio Code, in **TERMINAL**, start the console app using `dotnet watch`, and note the output that shows that hot reload is active, as shown in the following output:

```
dotnet watch 🔥 Hot reload enabled. For a list of supported edits, see https://aka.ms/dotnet/hot-
  💡 Press "Ctrl + R" to restart.
dotnet watch 🛠️ Building...
  Determining projects to restore...
  All projects are up-to-date for restore.
  HotReloading -> C:\cs11dotnet7\Chapter04\HotReloading\bin\Debug\net7.0\HotReloading.dll
dotnet watch 🚀 Started
Hello, Hot Reload!
Hello, Hot Reload!
Hello, Hot Reload!
```

1. In Visual Studio Code, change `Hello` to `Goodbye`, and note that after a couple of seconds the change is applied without needing to restart the console app, as shown in the following output:

```
Hello, Hot Reload!
dotnet watch ⌚ File changed: .\Program.cs.
```

```
Hello, Hot Reload!
Hello, Hot Reload!
dotnet watch 🔥 Hot reload of changes succeeded.
Goodbye, Hot Reload!
Goodbye, Hot Reload!
```

1. Press Ctrl + C to stop it running, as shown in the following output:

```
Goodbye, Hot Reload!
dotnet watch 🛑 Shutdown requested. Press Ctrl+C again to force exit.
```

Logging during development and runtime

Once you believe that all the bugs have been removed from your code, you would then compile a release version and deploy the application, so that people can use it. But no code is ever bug-free, and during runtime unexpected errors can occur.

End users are notoriously bad at remembering, admitting to, and then accurately describing what they were doing when an error occurred. You should not rely on them accurately providing useful information to reproduce the problem so that you can understand what caused the problem and then fix it. Instead, you can **instrument your code**, which means logging events of interest.

> **Good Practice**: Add code throughout your application to log what is happening, and especially when exceptions occur, so that you can review the logs and use them to trace the issue and fix the problem. Although we will see logging again in *Chapter 10*, *Working with Data Using Entity Framework Core*, and in *Chapter 14*, *Building Websites Using the Model-View-Controller Pattern*, logging is a huge topic, so we can only cover the basics in this book.

Understanding logging options

.NET includes some built-in ways to instrument your code by adding logging capabilities. We will cover the basics in this book. But logging is an area where third parties have created a rich ecosystem of powerful solutions that extend what Microsoft provides. I cannot make specific recommendations because the best logging framework depends on your needs. But I include some common ones in the following list:

- Apache log4net
- NLog
- Serilog

Instrumenting with Debug and Trace

There are two types that can be used to add simple logging to your code: `Debug` and `Trace`.

Before we delve into them in more detail, let's look at a quick overview of each one:

- The `Debug` class is used to add logging that gets written only during development.
- The `Trace` class is used to add logging that gets written during both development and runtime.

You have seen the use of the `Console` type and its `WriteLine` method writing out to the console window. There is also a pair of types named `Debug` and `Trace` that have more flexibility in where they write out to.

The `Debug` and `Trace` classes write to any trace listener. A trace listener is a type that can be configured to write output anywhere you like when the `WriteLine` method is called. There are several trace listeners provided by .NET, including one that outputs to the console, and you can even make your own by inheriting from the `TraceListener` type.

Writing to the default trace listener

One trace listener, the `DefaultTraceListener` class, is configured automatically and writes to Visual Studio Code's **DEBUG CONSOLE** window or Visual Studio's **Debug** window. You can configure other trace listeners using code.

Let's see trace listeners in action:

1. Use your preferred coding tool to add a new **Console App** / `console` project named `Instrumenting` to the `Chapter04` workspace/solution.
   - In Visual Studio Code, select `Instrumenting` as the active OmniSharp project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.
2. In `Program.cs`, delete the existing statements and then import the `System.Diagnostics` namespace, as shown in the following code:

```
using System.Diagnostics;
```

1. In `Program.cs`, write a message from the `Debug` and `Trace` classes, as shown in the following code:

```
Debug.WriteLine("Debug says, I am watching!");
Trace.WriteLine("Trace says, I am watching!");
```

1. In Visual Studio, navigate to **View** | **Output** and make sure **Show output from: Debug** is selected.
2. Start debugging the `Instrumenting` console app, and note that **DEBUG CONSOLE** in Visual Studio Code or the **Output** window in Visual Studio 2022 shows the two messages, mixed with other debugging information, such as loaded assembly DLLs, as shown in *Figures 4.12* and *4.13*:

Figure 4.12: Visual Studio Code DEBUG CONSOLE shows the two messages in blue

Figure 4.13: Visual Studio 2022 Output window shows Debug output including the two messages

Configuring trace listeners

Now, we will configure another trace listener that will write to a text file:

1. Before the `Debug` and `Trace` calls to `WriteLine`, add statements to create a new text file on the desktop and pass it into a new trace listener that knows how to write to a text file, and enable automatic flushing for its buffer, as shown in the following code:

```
string logPath = Path.Combine(Environment.GetFolderPath(
    Environment.SpecialFolder.DesktopDirectory), "log.txt");
Console.WriteLine($"Writing to: {logPath}");
TextWriterTraceListener logFile = new(File.CreateText(logPath));
Trace.Listeners.Add(logFile);
// text writer is buffered, so this option calls
// Flush() on all listeners after writing
Trace.AutoFlush = true;
```

Good Practice: Any type that represents a file usually implements a buffer to improve performance. Instead of writing immediately to the file, data is written to an in-memory buffer and only once the buffer is full will it be written in one chunk to the file. This behavior can be confusing while debugging because we do not immediately see the results! Enabling `AutoFlush` means the `Flush` method is called automatically after every write.

1. Run the release configuration of the console app:
   - In Visual Studio Code, enter the following command in the **TERMINAL** window for the `Instrumenting` project and note that nothing will appear to have happened:

```
dotnet run --configuration Release
```

- In Visual Studio 2022, in the standard toolbar, select **Release** in the **Solution Configurations** dropdown list and then navigate to **Debug** | **Start Without Debugging**, as shown in *Figure 4.14*:

Figure 4.14: Selecting the Release configuration in Visual Studio

1. On your desktop, open the file named `log.txt` and note that it contains the message `Trace says, I am watching!`.

2. Run the debug configuration of the console app:
    - In Visual Studio Code, enter the following command in the **TERMINAL** window for the `Instrumenting` project:

```
dotnet run --configuration Debug
```

- In Visual Studio, in the standard toolbar, select **Debug** in the **Solution Configurations** dropdown list and then navigate to **Debug** | **Start Debugging**.

1. On your desktop, open the file named `log.txt` and note that it contains both the message `Debug says, I am watching!` and also `Trace says, I am watching!`, as shown in *Figure 4.15*:

Figure 4.15: Opening the log.txt file in Visual Studio Code

Good Practice: When running with the `Debug` configuration, both `Debug` and `Trace` are active and will write to any trace listeners. When running with the `Release` configuration, only `Trace` will write to any trace listeners. You can therefore use `Debug.WriteLine` calls liberally throughout your code, knowing they will be stripped out automatically when you build the release version of your application and will therefore not affect performance.

Switching trace levels

The `Trace.WriteLine` calls are left in your code even after release. So, it would be great to have fine control over when they are output. This is something we can do with a **trace switch**.

The value of a trace switch can be set using a number or a word. For example, the number `3` can be replaced with the word `Info`, as shown in the following table:

Number	Word	Description
0	`Off`	This will output nothing.
1	`Error`	This will output only errors.
2	`Warning`	This will output errors and warnings.
3	`Info`	This will output errors, warnings, and information.
4	`Verbose`	This will output all levels.

Let's explore using trace switches. First, we will add some NuGet packages to our project to enable loading configuration settings from a JSON `appsettings` file.

Adding packages to a project in Visual Studio 2022

Visual Studio has a graphical user interface for adding packages.

1. In **Solution Explorer**, right-click the `Instrumenting` project and select **Manage NuGet Packages**.
2. Select the **Browse** tab.
3. In the search box, enter `Microsoft.Extensions.Configuration`.
4. Select each of these NuGet packages and click the **Install** button, as shown in *Figure 4.16*:
    - Microsoft.Extensions.Configuration
    - Microsoft.Extensions.Configuration.Binder
    - Microsoft.Extensions.Configuration.FileExtensions
    - Microsoft.Extensions.Configuration.Json

Figure 4.16: Installing NuGet packages using Visual Studio 2022

Good Practice: There are also packages for loading configuration from XML files, INI files, environment variables, and the command line. Use the most appropriate technique for setting configuration in your projects.

Adding packages to a project in Visual Studio Code

Visual Studio Code does not have a mechanism to add NuGet packages to a project, so we will use the command-line tool:

1. Navigate to the **TERMINAL** window for the `Instrumenting` project.
2. Enter the following command:

```
dotnet add package Microsoft.Extensions.Configuration
```

1. Enter the following command:

```
dotnet add package Microsoft.Extensions.Configuration.Binder
```

1. Enter the following command:

```
dotnet add package Microsoft.Extensions.Configuration.FileExtensions
```

1. Enter the following command:

```
dotnet add package Microsoft.Extensions.Configuration.Json
```

`dotnet add package` adds a reference to a NuGet package to your project file. It will be downloaded during the build process. `dotnet add reference` adds a project-to-project reference to your project file. The referenced project will be compiled if needed during the build process.

Reviewing project packages

After adding the NuGet packages, we can see the references in the project file:

1. Open `Instrumenting.csproj` and note the `<ItemGroup>` section with the added NuGet packages, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference
      Include="Microsoft.Extensions.Configuration"
      Version="7.0.0" />
    <PackageReference
      Include="Microsoft.Extensions.Configuration.Binder"
      Version="7.0.0" />
    <PackageReference
      Include="Microsoft.Extensions.Configuration.FileExtensions"
      Version="7.0.0" />
    <PackageReference
      Include="Microsoft.Extensions.Configuration.Json"
      Version="7.0.0" />
  </ItemGroup>
</Project>
```

Early Access Program: Until the final release of .NET 7, the version numbers of packages will be for previews and release candidates. For example, `7.0.0-preview.1.22109.13`. You can change the version to

`7.0.0-*` to automatically upgrade to the latest packages as monthly previews and release candidates are released.

1. Add a file named `appsettings.json` to the `Instrumenting` project folder.
2. In `appsettings.json`, define a setting named `PacktSwitch` with a `Level` value, as shown in the following code:

```
{
  "PacktSwitch": {
    "Level": "Info"
  }
}
```

1. In Visual Studio 2022, in **Solution Explorer**, right-click `appsettings.json`, select **Properties**, and then in the **Properties** window, change **Copy to Output Directory** to **Copy if newer**. This is necessary because unlike Visual Studio Code, which runs the console app in the project folder, Visual Studio runs the console app in `Instrumenting\bin\Debug\net7.0` or `Instrumenting\bin\Release\net7.0`.
2. In `Program.cs`, import the `Microsoft.Extensions.Configuration` namespace, as shown in the following code:

```
using Microsoft.Extensions.Configuration;
```

1. Add some statements to the end of `Program.cs` to create a configuration builder that looks in the current folder for a file named `appsettings.json`, build the configuration, create a trace switch, set its level by binding to the configuration, and then output the four trace switch levels, as shown in the following code:

```
Console.WriteLine("Reading from appsettings.json in {0}",
  arg0: Directory.GetCurrentDirectory());
ConfigurationBuilder builder = new();
builder.SetBasePath(Directory.GetCurrentDirectory());
builder.AddJsonFile("appsettings.json",
    optional: true, reloadOnChange: true);
IConfigurationRoot configuration = builder.Build();
TraceSwitch ts = new(
  displayName: "PacktSwitch",
  description: "This switch is set via a JSON config.");
configuration.GetSection("PacktSwitch").Bind(ts);
Trace.WriteLineIf(ts.TraceError, "Trace error");
Trace.WriteLineIf(ts.TraceWarning, "Trace warning");
Trace.WriteLineIf(ts.TraceInfo, "Trace information");
Trace.WriteLineIf(ts.TraceVerbose, "Trace verbose");
Console.ReadLine();
```

1. Set a breakpoint on the `Bind` statement.
2. Start debugging the `Instrumenting` console app.

3. In the **VARIABLES** or **Locals** window, expand the `ts` variable expression, and note that its `Level` is `Off` and its `TraceError`, `TraceWarning`, and so on are all `false`, as shown in *Figure 4.17*:

Figure 4.17: Watching the trace switch variable properties in Visual Studio 2022

4. Step into the call to the `Bind` method by clicking the **Step Into** or **Step Over** buttons or pressing F11 or F10, and note the `ts` variable watch expression updates to the `Info` level.

5. Step into or over the four calls to `Trace.WriteLineIf` and note that all levels up to `Info` are written to the **DEBUG CONSOLE** or **Output - Debug** window, but not `Verbose`, as shown in *Figure 4.18*:

Figure 4.18: Different trace levels shown in the DEBUG CONSOLE in Visual Studio Code

6. Stop debugging.
7. Modify `appsettings.json` to set a level of `2`, which means warning, as shown in the following JSON file:

```
{
  "PacktSwitch": {
    "Level": "2"
  }
}
```

1. Save the changes.
2. In Visual Studio Code, run the console application by entering the following command in the **TERMINAL** window for the `Instrumenting` project:

```
dotnet run --configuration Release
```

1. In Visual Studio, in the standard toolbar, select **Release** in the **Solution Configurations** dropdown list and then run the console app by navigating to **Debug | Start Without Debugging**.

2. Open the file named `log.txt` and note that this time, only trace error and warning levels are the output of the four potential trace levels, as shown in the following text file:

```
Trace says, I am watching!
Trace error
Trace warning
```

If no argument is passed, the default trace switch level is `Off` (0), so none of the switch levels are output.

Unit testing

Fixing bugs in code is expensive. The earlier that a bug is discovered in the development process, the less expensive it will be to fix.

Unit testing is a good way to find bugs early in the development process. Some developers even follow the principle that programmers should create unit tests before they write code, and this is called **Test-Driven Development** (**TDD**).

Microsoft has a proprietary unit testing framework known as **MS Test**. There is also a framework named **NUnit**. However, we will use the free and open-source third-party framework **xUnit.net**. xUnit was created by the same team that built NUnit, but they fixed the mistakes they felt they made previously. xUnit is more extensible and has better community support.

Understanding types of testing

Unit testing is just one of many types of testing, as described in the following table:

Type of testing	Description
Unit	Tests the smallest unit of code, typically a method or function. Unit testing is performed on a unit of code isolated from its dependencies by mocking them if needed. Each unit should have multiple tests: some with typical inputs and expected outputs, some with extreme input values to test boundaries, and some with deliberately wrong inputs to test exception handling.
Integration	Tests if the smaller units and larger components work together as a single piece of software. Sometimes involves integrating with external components that you do not have source code for.
System	Tests the whole system environment in which your software will run.
Performance	Tests the performance of your software; for example, your code must return a web page full of data to a visitor in under 20 milliseconds.
Load	Tests how many requests your software can handle simultaneously while maintaining required performance, for example, 10,000 concurrent visitors to a website.
User Acceptance	Tests if users can happily complete their work using your software.

Creating a class library that needs testing

First, we will create a function that needs testing. We will create it in a class library project separate from a console app project. A class library is a package of code that can be distributed and referenced by other .NET applications:

1. Use your preferred coding tool to add a new **Class Library** / `classlib` project named `CalculatorLib` to the `Chapter04` workspace/solution. At this point, you will have created about a dozen new console app projects and added them to a Visual Studio 2022 solution or a Visual Studio Code workspace. The only difference when adding a **Class Library** / `classlib` is to select a different project template. The rest of the steps are the same as adding a **Console App** / `console` project. For your convenience, I repeat them below for both Visual Studio 2022 and Visual Studio Code.

- If you are using Visual Studio 2022:

1. Navigate to **File** | **Add** | **New Project**.
2. In the **Add a new project** dialog, in **Recent project templates**, select **Class Library [C#]** and then click **Next**.
3. In the **Configure your new project** dialog, for the **Project name**, enter `CalculatorLib`, leave the location as `C:\cs11dotnet7\Chapter04`, and then click **Next**.
4. In the **Additional information** dialog, select **.NET 7.0 (Current)**, and then click **Create**.

- If you are using Visual Studio Code:

1. Navigate to **File** | **Add Folder to Workspace…**.
2. In the `Chapter04` folder, use the **New Folder** button to create a new folder named `CalculatorLib`, select it, and click **Add**.
3. When prompted if you trust the folder, click **Yes**.
4. Navigate to **Terminal** | **New Terminal**, and in the drop-down list that appears, select **CalculatorLib**.
5. In **TERMINAL**, confirm that you are in the `CalculatorLib` folder, and then enter the command to create a new class library, as shown in the following command: `dotnet new console`
6. Navigate to **View** | **Command Palette**, and then select **OmniSharp: Select Project**.
7. In the drop-down list, select the **CalculatorLib** project, and when prompted, click **Yes** to add required assets to debug.

1. In the `CalculatorLib` project, rename the file named `Class1.cs` to `Calculator.cs`.
2. Modify the file to define a `Calculator` class (with a deliberate bug!), as shown in the following code:

```
namespace CalculatorLib
{
  public class Calculator
  {
    public double Add(double a, double b)
    {
      return a * b;
    }
  }
}
```

1. Compile your class library project:
   - In Visual Studio 2022, navigate to **Build** | **Build CalculatorLib**.
   - In Visual Studio Code, in **TERMINAL**, enter the command `dotnet build`.
2. Use your preferred coding tool to add a new **xUnit Test Project [C#]** / `xunit` project named `CalculatorLibUnitTests` to the `Chapter04` workspace/solution.
3. Add a project reference to the `CalculatorLib` project:
   - If you are using Visual Studio 2022, in **Solution Explorer**, select the `CalculatorLibUnitTests` project, navigate to **Project** | **Add Project Reference…**, check the box to select the `CalculatorLib` project, and then click **OK**.
   - If you are using Visual Studio Code, use the `dotnet add reference` command or click on the file named `CalculatorLibUnitTests.csproj`, and modify the configuration to add an item group with a project reference to the `CalculatorLib` project, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <IsPackable>false</IsPackable>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.0.0" />
    <PackageReference Include="xunit" Version="2.4.1" />
    <PackageReference Include="xunit.runner.visualstudio" Version="2.4.3">
      <IncludeAssets>runtime; build; native; contentfiles;
        analyzers; buildtransitive</IncludeAssets>
```

```
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
    <PackageReference Include="coverlet.collector" Version="3.1.0">
      <IncludeAssets>runtime; build; native; contentfiles;
        analyzers; buildtransitive</IncludeAssets>
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
  </ItemGroup>
  <ItemGroup>
    <ProjectReference
      Include="..\CalculatorLib\CalculatorLib.csproj" />
  </ItemGroup>
</Project>
```

1. Build the `CalculatorLibUnitTests` project.

Writing unit tests

A well-written unit test will have three parts:

- **Arrange**: This part will declare and instantiate variables for input and output.
- **Act**: This part will execute the unit that you are testing. In our case, that means calling the method that we want to test.
- **Assert**: This part will make one or more assertions about the output. An assertion is a belief that, if not true, indicates a failed test. For example, when adding 2 and 2, we would expect the result to be 4.

Now, we will write some unit tests for the `Calculator` class:

1. Rename the file `UnitTest1.cs` to `CalculatorUnitTests.cs` and then open it.
2. In Visual Studio Code, rename the class to `CalculatorUnitTests`. (Visual Studio prompts you to rename the class when you rename the file.)
3. Import the `Packt` namespace.
4. Modify the `CalculatorUnitTests` class to have two test methods; one for adding 2 and 2, and another for adding 2 and 3, as shown in the following code:

```
using CalculatorLib;
namespace CalculatorLibUnitTests
{
  public class CalculatorUnitTests
  {
    [Fact]
    public void TestAdding2And2()
    {
      // arrange
      double a = 2;
      double b = 2;
      double expected = 4;
      Calculator calc = new();
      // act
      double actual = calc.Add(a, b);
      // assert
      Assert.Equal(expected, actual);
    }
    [Fact]
    public void TestAdding2And3()
    {
      // arrange
      double a = 2;
      double b = 3;
      double expected = 5;
      Calculator calc = new();
      // act
      double actual = calc.Add(a, b);
      // assert
      Assert.Equal(expected, actual);
    }
```

```
    }
}
```

Running unit tests using Visual Studio 2022

Now we are ready to run the unit tests and see the results:

1. In Visual Studio, navigate to **Test | Run All Tests**.
2. In **Test Explorer**, note that the results indicate that two tests ran, one test passed, and one test failed, as shown in *Figure 4.19*:

Figure 4.19: The unit test results in Visual Studio 2022's Test Explorer

Running unit tests using Visual Studio Code

Now we are ready to run the unit tests and see the results:

1. In Visual Studio Code, in the `CalculatorLibUnitTest` project's **TERMINAL** window, run the tests, as shown in the following command:

```
dotnet test
```

1. Note that the results indicate that two tests ran, one test passed, and one test failed, as shown in *Figure 4.20*:

Figure 4.20: The unit test results in Visual Studio Code's TERMINAL

Fix the bug

Now you can fix the bug:

1. Fix the bug in the `Add` method.
2. Run the unit tests again to see that the bug has now been fixed and both tests pass.

Throwing and catching exceptions in functions

In *Chapter 3*, *Controlling Flow, Converting Types, and Handling Exceptions*, you were introduced to exceptions and how to use a `try-catch` statement to handle them. But you should only catch and handle an exception if you have enough information to mitigate the issue. If you do not, then you should allow the exception to pass up through the call stack to a higher level.

Understanding usage errors and execution errors

Usage errors are when a programmer misuses a function, typically by passing invalid values as parameters. They could be avoided by that programmer changing their code to pass valid values. When some programmers first learn C# and .NET, they sometimes think exceptions can always be avoided because they assume all errors are usage errors. Usage errors should all be fixed before production runtime.

Execution errors are when something happens at runtime that cannot be fixed by writing "better" code. Execution errors can be split into **program errors** and **system errors**. If you attempt to access a network resource but the network is down, you need to be able to handle that system error by logging an exception, and possibly backing off for a time and trying again. But some system errors, such as running out of memory, simply cannot be handled. If

you attempt to open a file that does not exist, you might be able to catch that error and handle it programmatically by creating a new file. Program errors can be programmatically fixed by writing smart code. System errors often cannot be fixed programmatically.

Commonly thrown exceptions in functions

Very rarely should you define new types of exceptions to indicate usage errors. .NET already defines many that you should use.

When defining your own functions with parameters, your code should check the parameter values and throw exceptions if they have values that will prevent your function from properly functioning.

For example, if a parameter to a function should not be `null`, throw `ArgumentNullException`. For other problems, throw `ArgumentException`, `NotSupportedException`, or `InvalidOperationException`. For any exception, include a message that describes the problem for whoever will have to read it (typically a developer audience for class libraries and functions, or end users if it is at the highest level of a GUI app), as shown in the following code:

```
static void Withdraw(string accountName, decimal amount)
{
  if (accountName is null)
  {
    throw new ArgumentNullException(paramName: nameof(accountName));
  }
  if (amount < 0)
  {
    throw new ArgumentException(
      message: $"{nameof(amount)} cannot be less than zero.");
  }
  // process parameters
}
```

> **Good Practice**: If a function cannot successfully perform its operation, you should consider that a function failure and report it by throwing an exception.

Instead of writing an `if` statement and then throwing a new exception, .NET 6 introduced a convenience method to throw an exception if an argument is `null`, as shown in the following code:

```
static void Withdraw(string accountName, decimal amount)
{
  ArgumentNullException.ThrowIfNull(accountName);
```

C# 11 introduces a null check operator `!!` to do the same thing, as shown in the following code:

```
static void Withdraw(string accountName!!, decimal amount)
```

You should never need to write a `try-catch` statement to catch these usage type errors. You want the application to terminate. These exceptions should cause the programmer who is calling the function to fix their code to prevent the problem. They should be fixed before production deployment. That does not mean that your code does not need to throw usage error type exceptions. It should—to force other programmers to call your functions correctly!

Understanding the call stack

The entry point for a .NET console application is the `Main` (if you have explicitly defined this class) or `<Main>$` (if it was created for you by the top-level program feature) method of the `Program` class.

The `Main` method will call other methods, which call other methods, and so on, and these methods could be in the current project or in referenced projects and NuGet packages, as shown in *Figure 4.21*:

Figure 4.21: A chain of method calls that create a call stack

Let's create a similar chain of methods to explore where we could catch and handle exceptions:

1. Use your preferred coding tool to add a new **Class Library** / `classlib` project named `CallStackExceptionHandlingLib` to the `Chapter04` workspace/solution.
2. Rename the `Class1.cs` file to `Calculator.cs`.
3. In `Calculator.cs`, modify its contents, as shown in the following code:

```
using static System.Console;
namespace CallStackExceptionHandlingLib
{
  public class Calculator
  {
    public static void Gamma() // public so it can be called from outside
    {
      WriteLine("In Gamma");
      Delta();
    }
    private static void Delta() // private so it can only be called internally
    {
      WriteLine("In Delta");
      File.OpenText("bad file path");
    }
  }
}
```

1. Use your preferred coding tool to add a new **Console App** / `console` project named `CallStackExceptionHandling` to the `Chapter04` workspace/solution.
   - In Visual Studio Code, select `CallStackExceptionHandling` as the active OmniSharp project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.
2. In the `CallStackExceptionHandling` project, add a reference to the `CallStackExceptionHandlingLib` project.
3. In `Program.cs`, delete the existing statements and then add statements to define two methods and chain calls to them, and the methods in the class library, as shown in the following code:

```
using CallStackExceptionHandlingLib;
using static System.Console;
WriteLine("In Main");
Alpha();
void Alpha()
{
  WriteLine("In Alpha");
  Beta();
}
void Beta()
{
```

```
  WriteLine("In Beta");
  Calculator.Gamma();
}
```

1. Run the console app *without* the debugger attached, and note the results, as shown in the following partial output:

```
In Main
In Alpha
In Beta
In Gamma
In Delta
Unhandled exception. System.IO.FileNotFoundException: Could not find file 'C:\cs11dotnet7\Chapte
File name: 'C:\cs11dotnet7\Chapter04\CallStackExceptionHandling\bin\Debug\net7.0\bad file path'
   at Microsoft.Win32.SafeHandles.SafeFileHandle.CreateFile(String fullPath, FileMode mode, File
   at Microsoft.Win32.SafeHandles.SafeFileHandle.Open(String fullPath, FileMode mode, FileAccess
   at System.IO.Strategies.OSFileStreamStrategy..ctor(String path, FileMode mode, FileAccess acc
   at System.IO.Strategies.FileStreamHelpers.ChooseStrategyCore(String path, FileMode mode, File
   at System.IO.StreamReader.ValidateArgsAndOpenPath(String path, Encoding encoding, Int32 buffe
   at System.IO.File.OpenText(String path)
   at CallStackExceptionHandlingLib.Calculator.Delta() in C:\cs11dotnet7\Chapter04\CallStackExce
   at CallStackExceptionHandlingLib.Calculator.Gamma() in C:\cs11dotnet7\Chapter04\CallStackExce
   at Program.<<Main>$>g__Beta|0_1() in C:\cs11dotnet7\Chapter04\CallStackExceptionHandling\Prog
   at Program.<<Main>$>g__Alpha|0_0() in C:\cs11dotnet7\Chapter04\CallStackExceptionHandling\Pro
   at Program.<Main>$(String[] args) in C:\cs11dotnet7\Chapter04\CallStackExceptionHandling\Prog
```

Note the following:

- The call stack is upside-down. Starting from the bottom, you see:
  - The first call is to the `<Main>$` entry point function in the auto-generated `Program` class. This is where arguments are passed in as a `String` array.
  - The second call is to the `<<Main>$>g__Alpha|0_0` function. (The C# compiler renames it from `Alpha` when it adds it as a local function.)
  - The third call is to the `Beta` function.
  - The fourth call is to the `Gamma` function.
  - The fifth call is to the `Delta` function. This function attempts to open a file by passing a bad file path. This causes an exception to be thrown. Any function with a `try-catch` statement could catch this exception. If it does not, the exception is automatically passed up the call stack until it reaches the top, where .NET outputs the exception (and the details of this call stack).

Good Practice: Unless you need to step through your code to debug it, you should always run your code without the debugger attached. In this case, it is especially important not to attach the debugger because if you do, then it will catch the exception and show it in a GUI dialog box instead of outputting it as shown in the book.

Where to catch exceptions

Programmers can decide if they want to catch an exception near the failure point, or centralized higher up the call stack. This allows your code to be simplified and standardized. You might know that calling an exception could throw one or more types of exception, but you do not need to handle any of them at the current point in the call stack.

Rethrowing exceptions

Sometimes you want to catch an exception, log it, and then rethrow it. For example, if you are writing a low-level class library that will be called from an application, your code may not have enough information to programmatically fix the error in a smart way, but the calling application might have more information and could. Your code should log the error in case the calling application does not, and then rethrow it up the call stack in case the calling application chooses to handle it better.

There are three ways to rethrow an exception inside a `catch` block, as shown in the following list:

- To throw the caught exception with its original call stack, call `throw`.
- To throw the caught exception as if it was thrown at the current level in the call stack, call `throw` with the caught exception, for example, `throw ex`. This is usually poor practice because you have lost some potentially useful information for debugging but can be useful when you want to deliberately remove that information when it contains sensitive data.
- To wrap the caught exception in another exception that can include more information in a message that might help the caller understand the problem, throw a new exception, and pass the caught exception as the `innerException` parameter.

If an error could occur when we called the `Gamma` function, then we could catch the exception and then perform one of the three techniques of rethrowing an exception, as shown in the following code:

```
try
{
  Gamma();
}
catch (IOException ex)
{
  LogException(ex);
  // throw the caught exception as if it happened here
  // this will lose the original call stack
  throw ex;
  // rethrow the caught exception and retain its original call stack
  throw;
  // throw a new exception with the caught exception nested within it
  throw new InvalidOperationException(
    message: "Calculation had invalid values. See inner exception for why.",
    innerException: ex);
}
```

Let's see this in action with our call stack example:

1. In the `CallStackExceptionHandling` project, in `Program.cs`, in the `Beta` function, add a `try-catch` statement around the call to the `Gamma` function, as shown highlighted in the following code:

```
void Beta()
{
  WriteLine("In Beta");
  try
  {
    Calculator.Gamma();
  }
  catch (Exception ex)
  {
    WriteLine($"Caught this: {ex.Message}");
    throw ex;
  }
}
```

Note the green squiggle under the `throw ex` to warn you that you will lose call stack information.

1. Run the console app and note that the output excludes some details of the call stack, as shown in the following output:

```
Caught this: Could not find file 'C:\cs11dotnet7\Chapter04\CallStackExceptionHandling\bin\Debug\n
Unhandled exception. System.IO.FileNotFoundException: Could not find file 'C:\cs11dotnet7\Chapte
File name: 'C:\cs11dotnet7\Chapter04\CallStackExceptionHandling\bin\Debug\net7.0\bad file path'
   at Program.<<Main>$>g__Beta|0_1() in C:\cs11dotnet7\Chapter04\CallStackExceptionHandling\Prog
   at Program.<<Main>$>g__Alpha|0_0() in C:\cs11dotnet7\Chapter04\CallStackExceptionHandling\Pro
   at Program.<Main>$(String[] args) in C:\cs11dotnet7\Chapter04\CallStackExceptionHandling\Prog
```

1. Remove the `ex` by replacing the statement `throw ex;` with `throw;`.
2. Run the console app and note that the output includes all the details of the call stack.

Implementing the tester-doer pattern

The **tester-doer pattern** can avoid some thrown exceptions (but not eliminate them completely). This pattern uses pairs of functions: one to perform a test, the other to perform an action that would fail if the test is not passed.

.NET implements this pattern itself. For example, before adding an item to a collection by calling the `Add` method, you can test to see if it is read-only, which would cause `Add` to fail and therefore throw an exception.

For example, before withdrawing money from a bank account, you might test that the account is not overdrawn, as shown in the following code:

```
if (!bankAccount.IsOverdrawn())
{
  bankAccount.Withdraw(amount);
}
```

Problems with the tester-doer pattern

The tester-doer pattern can add performance overhead, so you can also implement the **try pattern**, which in effect combines the test and do parts into a single function, as we saw with `TryParse`.

Another problem with the tester-doer pattern occurs when you are using multiple threads. In this scenario, one thread could call the test function and it returns okay. But then another thread executes that changes the state. Then the original thread continues executing assuming everything is fine, but it is not fine. This is called a race condition. This topic is too advanced to cover how to handle in this book.

If you implement your own try pattern function and it fails, remember to set the `out` parameter to the default value of its type and then return `false`, as shown in the following code:

```
static bool TryParse(string? input, out Person value)
{
  if (someFailure)
  {
    value = default(Person);
    return false;
  }
  // successfully parsed the string into a Person
  value = new Person() { ... };
  return true;
}
```

Practicing and exploring

Test your knowledge and understanding by answering some questions, getting some hands-on practice, and exploring with deeper research into the topics covered in this chapter.

Exercise 4.1 – Test your knowledge

Answer the following questions. If you get stuck, try Googling the answers if necessary, while remembering that if you get totally stuck, the answers are in the Appendix:

1. What does the C# keyword `void` mean?
2. What are some differences between imperative and functional programming styles?
3. In Visual Studio Code or Visual Studio, what is the difference between pressing F5, Ctrl or Cmd + F5, Shift + F5, and Ctrl or Cmd + Shift + F5?
4. Where does the `Trace.WriteLine` method write its output to?
5. What are the five trace levels?
6. What is the difference between the `Debug` and `Trace` classes?
7. When writing a unit test, what are the three "A"s?
8. When writing a unit test using xUnit, what attribute must you decorate the test methods with?
9. What `dotnet` command executes xUnit tests?

10. What statement should you use to rethrow a caught exception named `ex` without losing the stack trace?

Exercise 4.2 – Practice writing functions with debugging and unit testing

Prime factors are the combination of the smallest prime numbers that, when multiplied together, will produce the original number. Consider the following example:

- Prime factors of 4 are: 2 x 2
- Prime factors of 7 are: 7
- Prime factors of 30 are: 5 x 3 x 2
- Prime factors of 40 are: 5 x 2 x 2 x 2
- Prime factors of 50 are: 5 x 5 x 2

Create three projects:

- A class library named `Ch04Ex02PrimeFactorsLib` with a static class and static method named `PrimeFactors` that, when passed an `int` variable as a parameter, returns a `string` showing its prime factors.
- A unit test project named `Ch04Ex02PrimeFactorsTests` with a few suitable unit tests.
- A console application to use it named `Ch04Ex02PrimeFactorsApp`.

To keep it simple, you can assume that the largest number entered will be 1,000.

Use the debugging tools and write unit tests to ensure that your function works correctly with multiple inputs and returns the correct output.

Exercise 4.3 – Explore topics

Use the links on the following page to learn more detail about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-4---writing-debugging-and-testing-functions

Summary

In this chapter, you learned:

- How to write reusable functions with input parameters and return values, in both an imperative and functional style.
- How to use the Visual Studio and Visual Studio Code debugging and diagnostic features like logging and unit tests to identify and fix any bugs in them.
- How to throw and catch exceptions in functions and understand the call stack.

In the next chapter, you will learn how to build your own types using object-oriented programming techniques.

5 Building Your Own Types with Object-Oriented Programming

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about making your own types using **object-oriented programming** (**OOP**). You will learn about all the different categories of members that a type can have, including fields to store data and methods to perform actions. You will use OOP concepts such as aggregation and encapsulation. You will also learn about language features such as tuple syntax support, out variables, inferred tuple names, and default literals. You will learn about defining operators and local functions for performing simple actions.

This chapter will cover the following topics:

- Talking about OOP
- Building class libraries
- Storing data within fields
- Writing and calling methods
- Splitting classes using partial
- Controlling access with properties and indexers
- More about methods
- Pattern matching with objects
- Working with records

Talking about OOP

An object in the real world is a thing, such as a car or a person, whereas an object in programming often represents something in the real world, such as a product or bank account, but it can also be something more abstract.

In C#, we use the `class` and `record` (mostly) or `struct` (sometimes) C# keywords to define a type of object. You will learn about the difference between classes and structs in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*. You can think of a type as being a blueprint or template for an object.

The concepts of OOP are briefly described here:

- **Encapsulation** is the combination of the data and actions that are related to an object. For example, a `BankAccount` type might have data, such as `Balance` and `AccountName`, as well as actions, such as `Deposit` and `Withdraw`. When encapsulating, you often want to control what can access those actions and the data; for example, restricting how the internal state of an object can be accessed or modified from the outside.
- **Composition** is about what an object is made of. For example, a `Car` is composed of different parts, such as four `Wheel` objects, several `Seat` objects, and an `Engine`.
- **Aggregation** is about what can be combined with an object. For example, a `Person` is not part of a `Car` object, but they could sit in the driver's `Seat` and then become the car's `Driver`—two separate objects that are aggregated together to form a new component.
- **Inheritance** is about reusing code by having a **subclass** derive from a **base** or **superclass**. All functionality in the base class is inherited by and becomes available in the **derived** class. For example, the base or super

`Exception` class has some members that have the same implementation across all exceptions, and the sub or derived `SqlException` class inherits those members and has extra members only relevant to when a SQL database exception occurs, like a property for the database connection.

- **Abstraction** is about capturing the core idea of an object and ignoring the details or specifics. C# has the `abstract` keyword that formalizes this concept. If a class is not explicitly **abstract**, then it can be described as being **concrete**. Base or superclasses are often abstract, for example, the superclass `Stream` is abstract, and its subclasses, like `FileStream` and `MemoryStream`, are concrete. Only concrete classes can be used to create objects; abstract classes can only be used as the base for other classes because they are missing some implementation. Abstraction is a tricky balance. If you make a class more abstract, more classes will be able to inherit from it, but at the same time, there will be less functionality to share.
- **Polymorphism** is about allowing a derived class to override an inherited action to provide custom behavior.

Building class libraries

Class library assemblies group types together into easily deployable units (DLL files). Apart from when you learned about unit testing, you have only created console applications or .NET Interactive notebooks to contain your code. To make the code that you write reusable across multiple projects, you should put it in class library assemblies, just like Microsoft does.

Creating a class library

The first task is to create a reusable .NET class library:

1. Use your preferred coding tool to create a new project, as defined in the following list:
   - Project template: **Class Library** `/ classlib`
   - Project file and folder: `PacktLibrary`
   - Workspace/solution file and folder: `Chapter05`
2. Open the `PacktLibrary.csproj` file, and note that, by default, class libraries created by the .NET SDK 7 target .NET 7 and therefore can only work with other .NET 7-compatible assemblies, as shown in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
</Project>
```

1. Modify the framework to target .NET Standard 2.0, add an entry to explicitly use the C# 11 compiler, and statically import the `System.Console` class to all C# files, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>netstandard2.0</TargetFramework>
    <LangVersion>11</LangVersion>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
  <ItemGroup>
    <Using Include="System.Console" Static="true" />
  </ItemGroup>
</Project>
```

Early Access Program: Until the final release of the C# 11 compiler, you will have to set `<LangVersion>` to `preview`.

1. Save and close the file.
2. Delete the file named `Class1.cs`.
3. Compile the project so that other projects can reference it later:

- In Visual Studio 2022, navigate to **Build** | **Build PacktLibrary**.
- In Visual Studio Code, enter the following command: `dotnet build`.

Good Practice: To use the latest C# language and .NET platform features, put types in a .NET 7 class library. To support legacy .NET platforms like .NET Core, .NET Framework, and Xamarin, put types that you might reuse in a .NET Standard 2.0 class library. By default, targeting .NET Standard 2.0 uses the C# 7.0 compiler, but this can be overridden so you get the benefits of the newer SDK and compiler even though you are limited to .NET Standard 2.0 APIs.

Defining a class in a namespace

The next task is to define a class that will represent a person:

1. In the `PacktLibrary` project, add a new class file named `Person.cs`.
2. Set the namespace to `Packt.Shared`, end the namespace definition with a semi-colon, and remove the curly braces associated with the namespace (Visual Studio 2022 does this automatically for you), to specify that the types defined in this file are part of this namespace, as shown in the following code:

```
namespace Packt.Shared; // file-scoped namespace
```

Good Practice: We're doing this because it is important to put your classes in a logically named namespace. A better namespace name would be domain-specific, for example, `System.Numerics` for types related to advanced numbers. In this case, the types we will create are `Person`, `BankAccount`, and `WondersOfTheWorld`, and they do not have a typical domain so we will use the more generic `Packt.Shared`.

1. For the `Person` class, change the keyword `internal` to `public`.

Your class file should now look like the following code:

```
namespace Packt.Shared; // file-scoped namespace
public class Person
{
}
```

Note that the C# keyword `public` is applied before `class`. This keyword is an **access modifier**, and it allows for any other code to access this class even outside this class library.

If you do not explicitly apply the `public` keyword, then it will only be accessible within the assembly that defined it. This is because the implicit access modifier for a class is `internal`. We need this class to be accessible outside the assembly, so we must make sure it is `public`.

To simplify your code if you are targeting .NET 6.0 or later and therefore using C# 10 or later, you can end a namespace declaration with a semi-colon and remove the curly braces, so the type definitions do not need to be indented. This is known as a **file-scoped namespace** declaration. You can only have one file-scoped namespace per file. This feature is especially useful for book writers who have limited horizontal space.

Good Practice: Put each type that you create in its own file so that you can use file-scoped namespace declarations.

Understanding members

This type does not yet have any members encapsulated within it. We will create some over the following pages. Members can be fields, methods, or specialized versions of both. You'll find a description of them here:

- **Fields** are used to store data. There are also three specialized categories of field, as shown in the following bullets:
  - **Constant**: The data never changes. The compiler literally copies the data into any code that reads it.

    - **Read-only**: The data cannot change after the class is instantiated, but the data can be calculated or loaded from an external source at the time of instantiation.
    - **Event**: The data references one or more methods that you want to execute when something happens, such as clicking on a button or responding to a request from some other code. Events will be covered in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*.
- **Methods** are used to execute statements. You saw some examples when you learned about functions in *Chapter 4*, *Writing, Debugging, and Testing Functions*. There are also four specialized categories of method:
    - **Constructor**: The statements execute when you use the `new` keyword to allocate memory to instantiate a class.
    - **Property**: The statements execute when you get or set data. The data is commonly stored in a field, but could be stored externally or calculated at runtime. Properties are the preferred way to encapsulate fields unless the memory address of the field needs to be exposed.
    - **Indexer**: The statements execute when you get or set data using "array" syntax `[]`.
    - **Operator**: The statements execute when you use an operator like `+` and `/` on operands of your type.

Instantiating a class

In this section, we will make an instance of the `Person` class.

Before we can instantiate a class, we need to reference the assembly that contains it from another project. We will use the class in a console app:

1. Use your preferred coding tool to add a new **Console App** / `console` named `PeopleApp` to the `Chapter05` workspace/solution.

If you are using Visual Studio 2022

1. Set the startup project for the solution to the current selection.
2. In **Solution Explorer**, select the `PeopleApp` project, navigate to **Project | Add Project Reference…**, check the box to select the `PacktLibrary` project, and then click **OK**.
3. In `PeopleApp.csproj`, add an entry to statically import the `System.Console` class, as shown in the following markup:

```
<ItemGroup>
  <Using Include="System.Console" Static="true" />
</ItemGroup>
```

1. Navigate to **Build** | **Build PeopleApp**.

If you are using Visual Studio Code

1. Select `PeopleApp` as the active `OmniSharp` project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.
2. Edit `PeopleApp.csproj` to add a project reference to `PacktLibrary`, and add an entry to statically import the `System.Console` class, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
  <ItemGroup>
    <ProjectReference Include="../PacktLibrary/PacktLibrary.csproj" />
  </ItemGroup>
  <ItemGroup>
    <Using Include="System.Console" Static="true" />
```

```
  </ItemGroup>
</Project>
```

1. In a terminal, compile the `PeopleApp` project and its dependency `PacktLibrary` project, as shown in the following command:

```
dotnet build
```

Importing a namespace to use a type

Now, we are ready to write statements to work with the `Person` class:

1. In the `PeopleApp` project/folder, in `Program.cs` file, delete the existing statements and then add statements to import the namespace for our `Person` class, as shown in the following code:

```
using Packt.Shared;
```

Although we could import this namespace globally, it will be clearer to anyone reading this code where we are importing the types we use from if the import statement is at the top of the file, and the `PeopleApp` project will only have this one `Program.cs` file that needs the namespace imported.

1. In `Program.cs`, add statements to:
   - Create an instance of the `Person` type.
   - Output the instance using a textual description of itself.

The `new` keyword allocates memory for the object and initializes any internal data, as shown in the following code:

```
// Person bob = new Person(); // C# 1.0 or later
// var bob = new Person(); // C# 3.0 or later
Person bob = new(); // C# 9.0 or later
WriteLine(bob.ToString());
```

You might be wondering, "Why does the `bob` variable have a method named `ToString`? The `Person` class is empty!" Don't worry, we're about to find out!

1. Run the code and view the result, as shown in the following output:

```
Packt.Shared.Person
```

Avoiding a namespace conflict with a using alias

It is possible that there are two namespaces that contain the same type name and importing both namespaces causes ambiguity. For example:

```
// France.Paris.cs
namespace France
{
  public class Paris
  {
  }
}
// Texas.Paris.cs
namespace France
{
  public class Paris
  {
  }
}
// Program.cs
using France;
using Texas;
Paris p = new();
```

The compiler would give `Error CS0104`:

```
'Paris' is an ambiguous reference between 'France.Paris' and 'Texas.Paris'
```

We can define an alias for one of the namespaces to differentiate it, as shown in the following code:

```
using France;
using us = Texas; // us becomes alias for the namespace and it is not imported
Paris p1 = new(); // France.Paris
us.Paris p2 = new(); // Texas.Paris
```

Renaming a type with a using alias

Another situation where you might want to use an alias is if you would like to rename a type. For example, if you use the `Environment` class a lot, you could rename it, as shown in the following code:

```
using Env = System.Environment;
WriteLine(Env.OSVersion);
WriteLine(Env.MachineName);
WriteLine(Env.CurrentDirectory);
```

Understanding objects

Although our `Person` class did not explicitly choose to inherit from a type, all types ultimately inherit directly or indirectly from a special type named `System.Object`.

The implementation of the `ToString` method in the `System.Object` type outputs the full namespace and type name.

Back in the original `Person` class, we could have explicitly told the compiler that `Person` inherits from the `System.Object` type, as shown in the following code:

```
public class Person : System.Object
```

When class B inherits from class A, we say that A is the base or superclass and B is the derived or subclass. In this case, `System.Object` is the base or superclass and `Person` is the derived or subclass.

You can also use the C# alias keyword `object`, as shown in the following code:

```
public class Person : object
```

Inheriting from System.Object

Let's make our class explicitly inherit from `object` and then review what members all objects have:

1. Modify your `Person` class to explicitly inherit from `object`.
2. Click inside the `object` keyword and press *F12*, or right-click on the `object` keyword and choose **Go to Definition**.

You will see the Microsoft-defined `System.Object` type and its members. This is something you don't need to understand the details of yet, but notice that it has a method named `ToString`, as shown in *Figure 5.1*:

Figure 5.1: System.Object class definition in .NET Standard 2.0

Good Practice: Assume other programmers know that if inheritance is not specified, the class will inherit from `System.Object`.

Storing data within fields

In this section, we will be defining a selection of fields in the class to store information about a person.

Defining fields

Let's say that we have decided that a person is composed of a name and a date of birth. We will encapsulate these two values inside a person, and the values will be visible outside it.

1. Inside the `Person` class, write statements to declare two public fields for storing a person's name and date of birth, as shown highlighted in the following code:

```
public class Person : object
{
  // fields
  public string? Name;
  public DateTime DateOfBirth;
}
```

We have multiple choices for the data type of the `DateOfBirth` field. .NET 6 introduced the `DateOnly` type. This would store only the date without a time value. `DateTime` stores the date and time of when the person

was born. An even better choice might be `DateTimeOffset`, which stores the date, time, and time zone. The choice depends on how much detail you need to store.

Since C# 8, the compiler has had the ability to warn you if a reference type like a `string` could have a `null` value and therefore potentially throw a `NullReferenceException`. Since .NET 6, the SDK enables those warnings by default. You can suffix the `string` type with a question mark `?` to indicate that you accept this, and the warning disappears. You will learn more about nullability and how to handle it in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*.

You can use any type for a field, including arrays and collections such as lists and dictionaries. These would be used if you needed to store multiple values in one named field. In this example, a person only has one name and one date of birth.

Understanding access modifiers

Part of encapsulation is choosing how visible the members are.

Note that, as we did with the class, we explicitly applied the `public` keyword to these fields. If we hadn't, then they would be implicitly `private` to the class, which means they are accessible only inside the class.

There are four access modifier keywords, and two combinations of access modifier keywords that you can apply to a class member, like a field or method, as shown in the following table:

Access Modifier	Description
`private`	Member is accessible inside the type only. This is the default.
`internal`	Member is accessible inside the type and any type in the same assembly.
`protected`	Member is accessible inside the type and any type that inherits from the type.
`public`	Member is accessible everywhere.
`internal` `protected`	Member is accessible inside the type, any type in the same assembly, and any type that inherits from the type. Equivalent to a fictional access modifier named `internal_or_protected`.
`private` `protected`	Member is accessible inside the type and any type that inherits from the type and is in the same assembly. Equivalent to a fictional access modifier named `internal_and_protected`. This combination is only available with C# 7.2 or later.

Good Practice: Explicitly apply one of the access modifiers to all type members, even if you want to use the implicit access modifier for members, which is `private`. Additionally, fields should usually be `private` or `protected`, and you should then create `public` properties to get or set the field values. This is because the property then controls access. You will do this later in the chapter.

Setting and outputting field values

Now we will use those fields in your code:

1. In `Program.cs`, after instantiating `bob`, add statements to set his name and date of birth, and then output those fields formatted nicely, as shown in the following code:

```
bob.Name = "Bob Smith";
bob.DateOfBirth = new DateTime(1965, 12, 22); // C# 1.0 or later
```

```
WriteLine(format: "{0} was born on {1:dddd, d MMMM yyyy}",
  arg0: bob.Name,
  arg1: bob.DateOfBirth);
```

We could have used string interpolation too, but for long strings it will wrap over multiple lines, which can be harder to read in a printed book. In the code examples in this book, remember that `{0}` is a placeholder for `arg0`, and so on.

1. Run the code and view the result, as shown in the following output:

```
Bob Smith was born on Wednesday, 22 December 1965
```

Your output may look different based on your locale, that is, language and culture.

The format code for `arg1` is made of several parts. `dddd` means the name of the day of the week. `d` means the number of the day of the month. `MMMM` means the name of the month. Lowercase `m` is used for minutes in time values. `yyyy` means the full number of the year. `yy` would mean the two-digit year.

You can also initialize fields using a shorthand **object initializer** syntax using curly braces that was introduced with C# 3.0. Let's see how.

1. Add statements underneath the existing code to create another new person named Alice. Note the different format code for the date of birth when writing her to the console, as shown in the following code:

```
Person alice = new()
{
  Name = "Alice Jones",
  DateOfBirth = new(1998, 3, 7) // C# 9.0 or later
};
WriteLine(format: "{0} was born on {1:dd MMM yy}",
  arg0: alice.Name,
  arg1: alice.DateOfBirth);
```

1. Run the code and view the result, as shown in the following output:

```
Alice Jones was born on 07 Mar 98
```

Storing a value using an enum type

Sometimes, a value needs to be one of a limited set of options. For example, there are seven ancient wonders of the world, and a person may have one favorite. At other times, a value needs to be a combination of a limited set of options. For example, a person may have a bucket list of ancient world wonders they want to visit. We are able to store this data by defining an `enum` type.

An `enum` type is a very efficient way of storing one or more choices because, internally, it uses integer values in combination with a lookup table of `string` descriptions:

1. Add a new file to the `PacktLibrary` project named `WondersOfTheAncientWorld.cs`.
2. Modify the `WondersOfTheAncientWorld.cs` file, as shown in the following code:

```
namespace Packt.Shared;
public enum WondersOfTheAncientWorld
{
  GreatPyramidOfGiza,
  HangingGardensOfBabylon,
  StatueOfZeusAtOlympia,
  TempleOfArtemisAtEphesus,
  MausoleumAtHalicarnassus,
  ColossusOfRhodes,
  LighthouseOfAlexandria
}
```

Good Practice: If you are writing code in a .NET Interactive notebook, then the code cell containing the `enum` must be above the code cell defining the `Person` class.

1. In `Person.cs`, add the following statement to your list of fields:

```
public WondersOfTheAncientWorld FavoriteAncientWonder;
```

1. In `Program.cs`, add the following statements:

```
bob.FavoriteAncientWonder = WondersOfTheAncientWorld.StatueOfZeusAtOlympia;
WriteLine(
  format: "{0}'s favorite wonder is {1}. Its integer is {2}.",
  arg0: bob.Name,
  arg1: bob.FavoriteAncientWonder,
  arg2: (int)bob.FavoriteAncientWonder);
```

1. Run the code and view the result, as shown in the following output:

```
Bob Smith's favorite wonder is StatueOfZeusAtOlympia. Its integer is 2.
```

The `enum` value is internally stored as an `int` for efficiency. The `int` values are automatically assigned starting at `0`, so the third world wonder in our `enum` has a value of `2`. You can assign `int` values that are not listed in the `enum`. They will output as the `int` value instead of a name, since a match will not be found.

Storing multiple values using an enum type

For the bucket list, we could create an array or collection of instances of the `enum`, and collections will be explained later in this chapter, but there is a better way. We can combine multiple choices into a single value using `enum` flags:

1. Modify the `enum` by decorating it with the `[Flags]` attribute, and explicitly set a `byte` value for each wonder which represents different bit columns, as shown in the following code:

```
namespace Packt.Shared;
[Flags]
public enum WondersOfTheAncientWorld : byte
{
  None                     = 0b_0000_0000, // i.e. 0
  GreatPyramidOfGiza       = 0b_0000_0001, // i.e. 1
  HangingGardensOfBabylon  = 0b_0000_0010, // i.e. 2
  StatueOfZeusAtOlympia    = 0b_0000_0100, // i.e. 4
  TempleOfArtemisAtEphesus = 0b_0000_1000, // i.e. 8
  MausoleumAtHalicarnassus = 0b_0001_0000, // i.e. 16
  ColossusOfRhodes         = 0b_0010_0000, // i.e. 32
  LighthouseOfAlexandria   = 0b_0100_0000  // i.e. 64
}
```

We are assigning explicit values for each choice that will not overlap when looking at the bits stored in memory. We should also decorate the `enum` type with the `System.Flags` attribute so that when the value is returned it can automatically match with multiple values as a comma-separated `string` instead of returning an `int` value.

Normally, an `enum` type uses an `int` variable internally, but since we don't need values that big, we can reduce memory requirements by 75%, that is, 1 byte per value instead of 4 bytes, by telling it to use a `byte` variable. As another example, if you wanted to define an `enum` for days of the week, there will only ever be seven of them.

If we want to indicate that our bucket list includes the *Hanging Gardens of Babylon* and the *Mausoleum at Halicarnassus* ancient world wonders, then we would want the `16` and `2` bits set to `1`. In other words, we would store the value `18`:

64 32 16 8 4 2 1
0 0 1 0 0 1 0

1. In `Person.cs`, add the following statement to your list of fields, as shown in the following code:

```
public WondersOfTheAncientWorld BucketList;
```

1. In `Program.cs`, add statements to set the bucket list using the `|` operator (bitwise logical OR) to combine the `enum` values. We could also set the value using the number `18` cast into the `enum` type, as shown in the comment, but we shouldn't because that would make the code harder to understand, as shown in the following code:

```
bob.BucketList =
  WondersOfTheAncientWorld.HangingGardensOfBabylon
  | WondersOfTheAncientWorld.MausoleumAtHalicarnassus;
// bob.BucketList = (WondersOfTheAncientWorld)18;
WriteLine($"{bob.Name}'s bucket list is {bob.BucketList}");
```

1. Run the code and view the result, as shown in the following output:

```
Bob Smith's bucket list is HangingGardensOfBabylon, MausoleumAtHalicarnassus
```

Good Practice: Use the `enum` values to store combinations of discrete options. Derive an `enum` type from `byte` if there are up to eight options, from `ushort` if there are up to 16 options, from `uint` if there are up to 32 options, and from `ulong` if there are up to 64 options.

Storing multiple values using collections

Let's now add a field to store a person's children. This is an example of aggregation because children are instances of a class that is related to the current person but are not part of the person itself. We will use a generic `List<T>` collection type that can store an ordered collection of any type. You will learn more about collections in *Chapter 8*, *Working with Common .NET Types*. For now, just follow along:

- In `Person.cs`, declare a new field to store multiple `Person` instances that represent the children of this person, as shown in the following code:

```
public List<Person> Children = new();
```

`List<Person>` is read aloud as "list of `Person`," for example, "the type of the property named `Children` is a list of `Person` instances."

We must ensure the collection is initialized to a new instance before we can add items to it, otherwise the field will be `null` and it will throw runtime exceptions when we try to use any of its members like `Add`.

Understanding generic collections

The angle brackets in the `List<T>` type is a feature of C# called **generics** that was introduced in 2005 with C# 2.0. It's a fancy term for making a collection **strongly typed**, that is, the compiler knows specifically what type of object can be stored in the collection. Generics improve the performance and correctness of your code.

Strongly typed has a different meaning to **statically typed**. The old `System.Collection` types are statically typed to contain weakly typed `System.Object` items. The newer `System.Collection.Generic` types are statically typed to contain strongly typed `<T>` instances.

Ironically, the term *generics* means we can use a more specific static type!

1. In `Program.cs`, add statements to add two children for `Bob` and then show how many children he has and what their names are, as shown in the following code:

```
bob.Children.Add(new Person { Name = "Alfred" }); // C# 3.0 and later
bob.Children.Add(new() { Name = "Zoe" }); // C# 9.0 and later
WriteLine($"{bob.Name} has {bob.Children.Count} children:");
for (int childIndex = 0; childIndex < bob.Children.Count; childIndex++)
```

```
{
  WriteLine($"> {bob.Children[childIndex].Name}");
}
```

We could also use a `foreach` statement to enumerate over the collection. As an extra challenge, change the `for` statement to output the same information using `foreach`.

1. Run the code and view the result, as shown in the following output:

```
Bob Smith has 2 children:
> Alfred
> Zoe
```

Making a field static

The fields that we have created so far have all been **instance** members, meaning that a different value of each field exists for each instance of the class that is created. The `alice` and `bob` variables have different `Name` values.

Sometimes, you want to define a field that only has one value that is shared across all instances.

These are called **static members** because fields are not the only members that can be static. Let's see what can be achieved using `static` fields:

1. In the `PacktLibrary` project, add a new class file named `BankAccount.cs`.
2. Modify the class to give it three fields, two instance fields and one static field, as shown in the following code:

```
namespace Packt.Shared;
public class BankAccount
{
  public string AccountName; // instance member
  public decimal Balance; // instance member
  public static decimal InterestRate; // shared member
}
```

Each instance of `BankAccount` will have its own `AccountName` and `Balance` values, but all instances will share a single `InterestRate` value.

1. In `Program.cs`, add statements to set the shared interest rate and then create two instances of the `BankAccount` type, as shown in the following code:

```
BankAccount.InterestRate = 0.012M; // store a shared value
BankAccount jonesAccount = new();
jonesAccount.AccountName = "Mrs. Jones";
jonesAccount.Balance = 2400;
WriteLine(format: "{0} earned {1:C} interest.",
  arg0: jonesAccount.AccountName,
  arg1: jonesAccount.Balance * BankAccount.InterestRate);
BankAccount gerrierAccount = new();
gerrierAccount.AccountName = "Ms. Gerrier";
gerrierAccount.Balance = 98;
WriteLine(format: "{0} earned {1:C} interest.",
  arg0: gerrierAccount.AccountName,
  arg1: gerrierAccount.Balance * BankAccount.InterestRate);
```

1. Run the code and view the additional output:

```
Mrs. Jones earned £28.80 interest.
Ms. Gerrier earned £1.18 interest.
```

`:C` is a format code that tells .NET to use the currency format for the numbers. It will use the default for your operating system installation. I live in London, UK, hence my output shows British Pounds (£).

You can control the culture that determines the currency symbol and other data formatting by setting a property on the current thread. For example, you could set the culture to British English near the top of `Program.cs`, as shown in the following code:

```
Thread.CurrentThread.CurrentCulture =
  System.Globalization.CultureInfo.GetCultureInfo("en-GB");
```

> If you would like to learn more about working with languages and cultures, as well as dates, times, and time zones, then there is a chapter about globalization and localization in my companion book, *Apps and Services with .NET 7*.

Fields are not the only members that can be static. Constructors, methods, properties, and other members can also be static.

Making a field constant

If the value of a field will never ever change, you can use the `const` keyword and assign a literal value at compile time:

1. In `Person.cs`, add a `string` constant for the species of a person, as shown in the following code:

```
 // constants
public const string Species = "Homo Sapiens";
```

1. To get the value of a constant field, you must write the name of the class, not the name of an instance of the class. In `Program.cs`, add a statement to write Bob's name and species to the console, as shown in the following code:

```
WriteLine($"{bob.Name} is a {Person.Species}");
```

1. Run the code and view the result, as shown in the following output:

```
Bob Smith is a Homo Sapiens
```

Examples of `const` fields in Microsoft types include `System.Int32.MaxValue` and `System.Math.PI` because neither value will ever change, as you can see in *Figure 5.2*:

Figure 5.2: Examples of constants in Math class

Good Practice: Constants are not always the best choice for two important reasons: the value must be known at compile time, and it must be expressible as a literal `string`, `Boolean`, or number value. Every reference to the `const` field is replaced with the literal value at compile time, which will, therefore, not be reflected if the value changes in a future version and you do not recompile any assemblies that reference it to get the new value.

Making a field read-only

Often, a better choice for fields that should not change is to mark them as read-only:

1. In `Person.cs`, add a statement to declare an instance read-only field to store a person's home planet, as shown in the following code:

```
// read-only fields
public readonly string HomePlanet = "Earth";
```

1. In `Program.cs`, add a statement to write Bob's name and home planet to the console, as shown in the following code:

```
WriteLine($"{bob.Name} was born on {bob.HomePlanet}");
```

1. Run the code and view the result, as shown in the following output:

```
Bob Smith was born on Earth
```

> **Good Practice**: Use read-only fields over constant fields for two important reasons: the value can be calculated or loaded at runtime and can be expressed using any executable statement. So, a read-only field can be set using a constructor or a field assignment. Every reference to the read-only field is a live reference, so any future changes will be correctly reflected by the calling code.

You can also declare `static readonly` fields whose values will be shared across all instances of the type.

Initializing fields with constructors

Fields often need to be initialized at runtime. You do this in a constructor that will be called when you make an instance of the class using the `new` keyword. Constructors execute before any fields are set by the code that is using the type.

1. In `Person.cs`, add statements after the existing read-only `HomePlanet` field to define a second read-only field and then set the `Name` and `Instantiated` fields in a constructor, as shown in the following code:

```
// read-only fields
public readonly string HomePlanet = "Earth";
public readonly DateTime Instantiated;
// constructors
public Person()
{
  // set default values for fields
  // including read-only fields
  Name = "Unknown";
  Instantiated = DateTime.Now;
}
```

1. In `Program.cs`, add statements to instantiate a new person and then output its initial field values, as shown in the following code:

```
Person blankPerson = new();
WriteLine(format:
  "{0} of {1} was created at {2:hh:mm:ss} on a {2:dddd}.",
  arg0: blankPerson.Name,
  arg1: blankPerson.HomePlanet,
  arg2: blankPerson.Instantiated);
```

1. Run the code and view the result, as shown in the following output:

```
Unknown of Earth was created at 11:58:12 on a Sunday
```

Defining multiple constructors

You can have multiple constructors in a type. This is especially useful to encourage developers to set initial values for fields:

1. In `Person.cs`, add statements to define a second constructor that allows a developer to set initial values for the person's name and home planet, as shown in the following code:

```
public Person(string initialName, string homePlanet)
{
  Name = initialName;
  HomePlanet = homePlanet;
  Instantiated = DateTime.Now;
}
```

1. In `Program.cs`, add statements to create another person using the constructor with two parameters, as shown in the following code:

```
Person gunny = new(initialName: "Gunny", homePlanet: "Mars");
WriteLine(format:
  "{0} of {1} was created at {2:hh:mm:ss} on a {2:dddd}.",
  arg0: gunny.Name,
  arg1: gunny.HomePlanet,
  arg2: gunny.Instantiated);
```

1. Run the code and view the result:

```
Gunny of Mars was created at 11:59:25 on a Sunday
```

Constructors are a special category of method. Let's look at methods in more detail.

Writing and calling methods

Methods are members of a type that execute a block of statements. They are functions that belong to a type.

Returning values from methods

Methods can return a single value or return nothing:

- A method that performs some actions but does not return a value indicates this with the `void` type before the name of the method.
- A method that performs some actions and returns a value indicates this with the type of the return value before the name of the method.

For example, in the next task, you will create two methods:

- `WriteToConsole` : This will perform an action (writing some text to the console), but it will return nothing from the method, indicated by the `void` keyword.
- `GetOrigin` : This will return a text value, indicated by the `string` keyword.

Let's write the code:

1. In `Person.cs` , add statements to define the two methods that I described earlier, as shown in the following code:

```
// methods
public void WriteToConsole()
{
  WriteLine($"{Name} was born on a {DateOfBirth:dddd}.");
}
public string GetOrigin()
{
  return $"{Name} was born on {HomePlanet}.";
}
```

1. In `Program.cs` , add statements to call the two methods, as shown in the following code:

```
bob.WriteToConsole();
WriteLine(bob.GetOrigin());
```

1. Run the code and view the result, as shown in the following output:

```
Bob Smith was born on a Wednesday.
Bob Smith was born on Earth.
```

Combining multiple returned values using tuples

Each method can only return a single value that has a single type. That type could be a simple type, such as `string` in the previous example; a complex type, such as `Person` ; or a collection type, such as `List<Person>` .

Imagine that we want to define a method named `GetTheData` that needs to return both a `string` value and an `int` value. We could define a new class named `TextAndNumber` with a `string` field and an `int` field, and return an instance of that complex type, as shown in the following code:

```
public class TextAndNumber
{
  public string Text;
  public int Number;
}
public class LifeTheUniverseAndEverything
{
  public TextAndNumber GetTheData()
  {
    return new TextAndNumber
    {
      Text = "What's the meaning of life?",
      Number = 42
    };
  }
}
```

But defining a class just to combine two values together is unnecessary, because in modern versions of C# we can use **tuples**. Tuples are an efficient way to combine two or more values into a single unit. I pronounce them as tuh-ples but I have heard other developers pronounce them as too-ples. To-may-toe, to-mah-toe, po-tay-toe, po-tah-toe, I guess.

Tuples have been a part of some languages such as F# since their first version, but .NET only added support for them with .NET 4.0 in 2010 using the `System.Tuple` type.

C# language support for tuples

It was only with C# 7.0 in 2017 that C# added language syntax support for tuples using the parentheses characters `()` and, at the same time, .NET added a new `System.ValueTuple` type that is more efficient in some common scenarios than the old .NET 4.0 `System.Tuple` type. The C# tuple syntax uses the more efficient one.

Let's explore tuples:

1. In `Person.cs`, add statements to define a method that returns a tuple that combines a `string` and `int`, as shown in the following code:

```
public (string, int) GetFruit()
{
  return ("Apples", 5);
}
```

1. In `Program.cs`, add statements to call the `GetFruit` method and then output the tuple's fields, which are automatically named `Item1` and `Item2`, as shown in the following code:

```
(string, int) fruit = bob.GetFruit();
WriteLine($"{fruit.Item1}, {fruit.Item2} there are.");
```

1. Run the code and view the result, as shown in the following output:

```
Apples, 5 there are.
```

Naming the fields of a tuple

To access the fields of a tuple, the default names are `Item1`, `Item2`, and so on.

You can explicitly specify the field names:

1. In `Person.cs`, add statements to define a method that returns a tuple with named fields, as shown in the following code:

```
public (string Name, int Number) GetNamedFruit()
{
  return (Name: "Apples", Number: 5);
}
```

1. In `Program.cs`, add statements to call the method and output the tuple's named fields, as shown in the following code:

```
var fruitNamed = bob.GetNamedFruit();
WriteLine($"There are {fruitNamed.Number} {fruitNamed.Name}.");
```

1. Run the code and view the result, as shown in the following output:

```
There are 5 Apples.
```

If you are constructing a tuple from another object, you can use a feature introduced in C# 7.1 called **tuple name inference**.

1. In `Program.cs`, create two tuples, made of a `string` and `int` value each, as shown in the following code:

```
var thing1 = ("Neville", 4);
WriteLine($"{thing1.Item1} has {thing1.Item2} children.");
var thing2 = (bob.Name, bob.Children.Count);
WriteLine($"{thing2.Name} has {thing2.Count} children.");
```

In C# 7.0, both things would use the `Item1` and `Item2` naming schemes. In C# 7.1 and later, `thing2` can infer the names `Name` and `Count`.

Deconstructing tuples

You can also deconstruct tuples into separate variables. The deconstructing declaration has the same syntax as named field tuples, but without a named variable for the tuple, as shown in the following code:

```
// store return value in a tuple variable with two fields
(string TheName, int TheNumber) tupleWithNamedFields = bob.GetNamedFruit();
// tupleWithNamedFields.TheName
// tupleWithNamedFields.TheNumber
// deconstruct return value into two separate variables
(string name, int number) = bob.GetNamedFruit();
// name
// number
```

This has the effect of splitting the tuple into its parts and assigning those parts to new variables.

1. In `Program.cs`, add statements to deconstruct the tuple returned from the `GetFruit` method, as shown in the following code:

```
(string fruitName, int fruitNumber) = bob.GetFruit();
WriteLine($"Deconstructed: {fruitName}, {fruitNumber}");
```

1. Run the code and view the result, as shown in the following output:

```
Deconstructed: Apples, 5
```

Deconstructing types

Tuples are not the only type that can be deconstructed. Any type can have special methods named `Deconstruct` that break down the object into parts. Let's implement some for the `Person` class:

1. In `Person.cs`, add two `Deconstruct` methods with `out` parameters defined for the parts we want to deconstruct into, as shown in the following code:

```
// deconstructors
public void Deconstruct(out string? name, out DateTime dob)
{
  name = Name;
  dob = DateOfBirth;
}
public void Deconstruct(out string? name,
  out DateTime dob, out WondersOfTheAncientWorld fav)
{
  name = Name;
  dob = DateOfBirth;
  fav = FavoriteAncientWonder;
}
```

Although I introduce the `out` keyword in the preceding code, you will learn about them properly in a few pages time.

1. In `Program.cs`, add statements to deconstruct `bob`, as shown in the following code:

```
// Deconstructing a Person
var (name1, dob1) = bob; // implicitly calls the Deconstruct method
WriteLine($"Deconstructed: {name1}, {dob1}");
var (name2, dob2, fav2) = bob;
WriteLine($"Deconstructed: {name2}, {dob2}, {fav2}");
```

You do not explicitly call the `Deconstruct` method. It is called implicitly when you assign an object to a tuple variable.

1. Run the code and view the result, as shown in the following output:

```
Deconstructed: Bob Smith, 22/12/1965 00:00:00
Deconstructed: Bob Smith, 22/12/1965 00:00:00, StatueOfZeusAtOlympia
```

Defining and passing parameters to methods

Methods can have parameters passed to them to change their behavior. Parameters are defined a bit like variable declarations but inside the parentheses of the method, as you saw earlier in this chapter with constructors. Let's see more examples:

1. In `Person.cs`, add statements to define two methods, the first without parameters and the second with one parameter, as shown in the following code:

```
public string SayHello()
{
  return $"{Name} says 'Hello!'";
}
public string SayHelloTo(string name)
{
  return $"{Name} says 'Hello, {name}!'";
}
```

1. In `Program.cs`, add statements to call the two methods and write the return value to the console, as shown in the following code:

```
WriteLine(bob.SayHello());
WriteLine(bob.SayHelloTo("Emily"));
```

1. Run the code and view the result:

```
Bob Smith says 'Hello!'
Bob Smith says 'Hello, Emily!'
```

When typing a statement that calls a method, IntelliSense shows a tooltip with the name and type of any parameters, and the return type of the method.

Overloading methods

Instead of having two different method names, we could give both methods the same name. This is allowed because the methods each have a different signature. You saw an example of this with the two `Deconstruct` methods.

A **method signature** is a list of parameter types that can be passed when calling the method. Overloaded methods cannot differ only in the return type.

1. In `Person.cs`, change the name of the `SayHelloTo` method to `SayHello`.
2. In `Program.cs`, change the method call to use the `SayHello` method, and note that the quick info for the method tells you that it has an additional overload, **1 of 2**, as well as **2 of 2**, as shown in *Figure 5.3*:

Figure 5.3: An IntelliSense tooltip for an overloaded method

Good Practice: Use overloaded methods to simplify your class by making it appear to have fewer methods.

Passing optional and named parameters

Another way to simplify methods is to make parameters optional. You make a parameter optional by assigning a default value inside the method parameter list. Optional parameters must always come last in the list of parameters.

We will now create a method with three optional parameters:

1. In `Person.cs`, add statements to define the method, as shown in the following code:

```
public string OptionalParameters(string command  = "Run!",
  double number = 0.0, bool active = true)
{
  return string.Format(
    format: "command is {0}, number is {1}, active is {2}",
    arg0: command,
    arg1: number,
```

```
    arg2: active);
}
```

1. In `Program.cs`, add a statement to call the method and write its return value to the console, as shown in the following code:

```
WriteLine(bob.OptionalParameters());
```

1. Watch IntelliSense appear as you type the code. You will see a tooltip, showing the three optional parameters with their default values, as shown in *Figure 5.4*:

Figure 5.4: IntelliSense showing optional parameters as you type code

2. Run the code and view the result, as shown in the following output:

```
command is Run!, number is 0, active is True
```

1. In `Program.cs`, add a statement to pass a `string` value for the `command` parameter and a `double` value for the `number` parameter, as shown in the following code:

```
WriteLine(bob.OptionalParameters("Jump!", 98.5));
```

1. Run the code and see the result, as shown in the following output:

```
command is Jump!, number is 98.5, active is True
```

The default values for the `command` and `number` parameters have been replaced, but the default for `active` is still `true`.

Naming parameter values when calling methods

Optional parameters are often combined with naming parameters when you call the method, because naming a parameter allows the values to be passed in a different order than how they were declared.

1. In `Program.cs`, add a statement to pass a `string` value for the `command` parameter and a `double` value for the `number` parameter but using named parameters, so that the order they are passed through can be swapped around, as shown in the following code:

```
WriteLine(bob.OptionalParameters(number: 52.7, command: "Hide!"));
```

1. Run the code and view the result, as shown in the following output:

```
command is Hide!, number is 52.7, active is True
```

You can even use named parameters to skip over optional parameters.

1. In `Program.cs`, add a statement to pass a `string` value for the `command` parameter using positional order, skip the `number` parameter, and use the named `active` parameter, as shown in the following code:

```
WriteLine(bob.OptionalParameters("Poke!", active: false));
```

1. Run the code and view the result, as shown in the following output:

```
command is Poke!, number is 0, active is False
```

Controlling how parameters are passed

When a parameter is passed into a method, it can be passed in one of three ways:

- By **value** (this is the default): Think of these as being *in-only*.
- By **reference** as a `ref` parameter: Think of these as being *in-and-out*.
- As an `out` parameter: Think of these as being *out-only*.

Let's see some examples of passing parameters in and out:

1. In `Person.cs`, add statements to define a method with three parameters, one `in` parameter, one `ref` parameter, and one `out` parameter, as shown in the following method:

```
public void PassingParameters(int x, ref int y, out int z)
{
  // out parameters cannot have a default
  // AND must be initialized inside the method
  z = 99;
  // increment each parameter
  x++;
  y++;
  z++;
}
```

1. In `Program.cs`, add statements to declare some `int` variables and pass them into the method, as shown in the following code:

```
int a = 10;
int b = 20;
int c = 30;
WriteLine($"Before: a = {a}, b = {b}, c = {c}");
bob.PassingParameters(a, ref b, out c);
WriteLine($"After: a = {a}, b = {b}, c = {c}");
```

1. Run the code and view the result, as shown in the following output:

```
Before: a = 10, b = 20, c = 30
After: a = 10, b = 21, c = 100
```

- When passing a variable as a parameter by default, its current value gets passed, not the variable itself. Therefore, `x` has a copy of the value of the `a` variable. The `a` variable retains its original value of `10`.

- When passing a variable as a `ref` parameter, a reference to the variable gets passed into the method. Therefore, `y` is a reference to `b`. The `b` variable gets incremented when the `y` parameter gets incremented.
- When passing a variable as an `out` parameter, a reference to the variable gets passed into the method. Therefore, `z` is a reference to `c`. The value of the `c` variable gets replaced by whatever code executes inside the method. We could simplify the code in the `Main` method by not assigning the value `30` to the `c` variable, since it will always be replaced anyway.

Simplified out parameters

In C# 7.0 and later, we can simplify code that uses the `out` parameter.

1. In `Program.cs`, add statements to declare some more variables, including an `out` parameter named `f` declared inline, as shown in the following code:

```
int d = 10;
int e = 20;
WriteLine($"Before: d = {d}, e = {e}, f doesn't exist yet!");
// simplified C# 7.0 or later syntax for the out parameter
bob.PassingParameters(d, ref e, out int f);
WriteLine($"After: d = {d}, e = {e}, f = {f}");
```

1. Run the code and view the result, as shown in the following output:

```
Before: d = 10, e = 20, f doesn't exist yet!
After: d = 10, e = 21, f = 100
```

Understanding ref returns

In C# 7.0 or later, the `ref` keyword is not just for passing parameters into a method; it can also be applied to the `return` value. This allows an external variable to reference an internal variable and modify its value after the method call. This might be useful in advanced scenarios, for example, passing around placeholders into big data structures, but it's beyond the scope of this book.

Splitting classes using partial

When working on large projects with multiple team members, or when working with especially large and complex class implementations, it is useful to be able to split the definition of a class across multiple files. You do this using the `partial` keyword.

Imagine we want to add statements to the `Person` class that are automatically generated by a tool like an object-relational mapper that reads schema information from a database. If the class is defined as `partial`, then we can split the class into an autogenerated code file and a manually edited code file.

Let's write some code that simulates this example:

1. In `Person.cs`, add the `partial` keyword, as shown highlighted in the following code:

```
public partial class Person
```

1. In the `PacktLibrary` project/folder, add a new class file named `PersonAutoGen.cs`.
2. Add statements to the new file, as shown in the following code:

```
namespace Packt.Shared;
// this file simulates an autogenerated class
public partial class Person
{
}
```

The rest of the code we write for this chapter will be written in the `PersonAutoGen.cs` file.

Controlling access with properties and indexers

Earlier, you created a method named `GetOrigin` that returned a `string` containing the name and origin of the person. Languages such as Java do this a lot. C# has a better way: properties.

A **property** is simply a method (or a pair of methods) that acts and looks like a field when you want to get or set a value, thereby simplifying the syntax.

Defining read-only properties

A `readonly` property only has a `get` implementation.

1. In `PersonAutoGen.cs`, in the `Person` class, add statements to define three properties:
   - The first property will perform the same role as the `GetOrigin` method using the property syntax that works with all versions of C#.
   - The second property will return a greeting message using the lambda expression body `=>` syntax from C# 6 and later.
   - The third property will calculate the person's age.
2. Here's the code:

```
// a readonly property defined using C# 1 - 5 syntax
public string Origin
{
  get
  {
    return string.Format("{0} was born on {1}",
      arg0: Name, arg1: HomePlanet);
  }
}
// two readonly properties defined using C# 6+ lambda expression body syntax
public string Greeting => $"{Name} says 'Hello!'";
public int Age => DateTime.Today.Year - DateOfBirth.Year;
```

Good Practice: This isn't the best way to calculate someone's age, but we aren't learning how to calculate an age from a date of birth. If you need to do that properly, read the discussion at the following link: https://stackoverflow.com/questions/9/how-do-i-calculate-someones-age-in-c

1. In `Program.cs`, add the statements to get the properties, as shown in the following code:

```
Person sam = new()
{
  Name = "Sam",
  DateOfBirth = new(1969, 6, 25)
};
WriteLine(sam.Origin);
WriteLine(sam.Greeting);
WriteLine(sam.Age);
```

1. Run the code and view the result, as shown in the following output:

```
Sam was born on Earth
Sam says 'Hello!'
53
```

The output shows 53 because I ran the console app on February 20, 2022, when Sam was 53 years old.

Defining settable properties

To create a settable property, you must use the older syntax and provide a pair of methods—not just a `get` part, but also a `set` part:

1. In `PersonAutoGen.cs`, add statements to define a `string` property that has both a `get` and `set` method (also known as a **getter** and **setter**), as shown in the following code:

```
// a read-write property defined using C# 3.0 syntax
public string? FavoriteIceCream { get; set; } // auto-syntax
```

Although you have not manually created a field to store the person's favorite ice cream, it is there, automatically created by the compiler for you.

Sometimes, you need more control over what happens when a property is set. In this scenario, you must use a more detailed syntax and manually create a `private` field to store the value for the property.

1. In `PersonAutoGen.cs`, add statements to define a `string` field and `string` property that has both a `get` and `set`, as shown in the following code:

```
// a private field to store the property value
private string? favoritePrimaryColor;
// a public property to read and write to the field
public string? FavoritePrimaryColor
{
  get
  {
    return favoritePrimaryColor;
  }
  set
  {
    switch (value?.ToLower())
    {
      case "red":
      case "green":
      case "blue":
        favoritePrimaryColor = value;
        break;
      default:
        throw new ArgumentException(
          $"{value} is not a primary color. " +
          "Choose from: red, green, blue.");
    }
  }
}
```

Good Practice: Avoid adding too much code to your getters and setters. This could indicate a problem with your design. Consider adding private methods that you then call in setters and getters to simplify your implementations.

1. In `Program.cs`, add statements to set Sam's favorite ice cream and color, and then write them out, as shown in the following code:

```
sam.FavoriteIceCream = "Chocolate Fudge";
WriteLine($"Sam's favorite ice-cream flavor is {sam.FavoriteIceCream}.");
sam.FavoritePrimaryColor = "Red";
WriteLine($"Sam's favorite primary color is {sam.FavoritePrimaryColor}.");
```

1. Run the code and view the result, as shown in the following output:

```
Sam's favorite ice-cream flavor is Chocolate Fudge.
Sam's favorite primary color is Red.
```

If you try to set the color to any value other than red, green, or blue, then the code will throw an exception. The calling code could then use a `try` statement to display the error message.

Good Practice: Use properties instead of fields when you want to read and write to a field without using a method pair like `GetAge` and `SetAge`.

Accessing the autogenerated private field

C# 11 introduces the `field` keyword. For autogenerated properties, a hidden private field is defined for you, but because it is hidden you could not refer to it. Until now.

Requiring properties to be set during instantiation

C# 11 introduces the `required` modifier. If you use it on a property, the compiler will ensure that you set the property to a value when you instantiate it.

1. In the `PacktLibrary` project, add a new class file named `Book.cs`.
2. Modify the class to give it two fields, one required, as shown in the following code:

```
namespace Packt.Shared;
public class Book
{
  public required string? Isbn { get; set; }
  public string? Title { get; set; }
}
```

1. In `Program.cs`, attempt to instantiate a `Book` without setting the `Isbn` property, as shown in the following code:

```
Book book = new();
book.Title = "C# 11 and .NET 7 - Modern Cross-Platform Development";
```

1. Note that you will see a compiler error, as shown in the following output:

Defining indexers

Indexers allow the calling code to use the array syntax to access a property. For example, the `string` type defines an indexer so that the calling code can access individual characters in the `string`, as shown in the following code:

```
string alphabet = "abcdefghijklmnopqrstuvwxyz";
char letterF = alphabet[5]; // 0 is a, 1 is b, and so on
```

We will define an indexer to simplify access to the children of a person:

1. In `PersonAutoGen.cs`, add statements to define an indexer to get and set a child using the index of the child, as shown in the following code:

```
// indexers
public Person this[int index]
{
  get
  {
    return Children[index]; // pass on to the List<T> indexer
  }
  set
  {
    Children[index] = value;
  }
}
```

You can overload indexers so that different types can be used for their parameters. For example, as well as passing an `int` value, you could also pass a `string` value.

1. In `PersonAutoGen.cs`, add statements to define an indexer to get and set a child using the name of the child, as shown in the following code:

```
public Person this[string name]
{
```

```
  get
  {
    return Children.Find(p => p.Name == name);
  }
  set
  {
    Person found = Children.Find(p => p.Name == name);
    if (found is not null) found = value;
  }
}
```

You will learn more about collections like `List<T>` in *Chapter 8*, *Working with Common .NET Types* and how to write lambda expressions using `=>` in *Chapter 11*, *Querying and Manipulating Data Using LINQ*.

1. In `Program.cs`, add statements to add two children to `Sam`, and then access the first and second child using the longer `Children` field and the shorter indexer syntax, as shown in the following code:

```
sam.Children.Add(new() { Name = "Charlie", DateOfBirth = new(2010, 3, 18) });
sam.Children.Add(new() { Name = "Ella", DateOfBirth = new(2020, 12, 24) });
// get using Children list
WriteLine($"Sam's first child is {sam.Children[0].Name}.");
WriteLine($"Sam's second child is {sam.Children[1].Name}.");
// get using integer position indexer
WriteLine($"Sam's first child is {sam[0].Name}.");
WriteLine($"Sam's second child is {sam[1].Name}.");
// get using name indexer
WriteLine($"Sam's child named Ella is {sam["Ella"].Age} years old.");
```

1. Run the code and view the result, as shown in the following output:

```
Sam's first child is Charlie.
Sam's second child is Ella.
Sam's first child is Charlie.
Sam's second child is Ella.
Sam's child named Ella is 2 years old.
```

More about methods

I wanted to think of some methods that would apply to a `Person` instance that could also become operators like `+` and `*`. What would adding two people together represent? What would multiplying two people represent? The obvious answers are getting married and making babies.

We might want two instances of `Person` to be able to marry and procreate. We can implement this by writing methods and overriding operators. Instance methods are actions that an object does to itself; static methods are actions the type does.

Which you choose depends on what makes the most sense for the action.

Good Practice: Having both static and instance methods to perform similar actions often makes sense. For example, `string` has both a `Compare` static method and a `CompareTo` instance method. This puts the choice of how to use the functionality in the hands of the programmers using your type, giving them more flexibility.

Implementing functionality using methods

Let's start by implementing some functionality by using both static and instance methods:

1. In `PersonAutoGen.cs`, in the `Person` class, add read-only properties with private backing storage fields to indicate if that person is married and to whom, as shown in the following code:

```
private bool married = false;
public bool Married => married;
private Person? spouse = null;
public Person? Spouse => spouse;
```

1. In `PersonAutoGen.cs`, add one instance method and one static method to the `Person` class that will allow two `Person` objects to marry, as shown in the following code:

```
// static method to marry
public static void Marry(Person p1, Person p2)
{
  p1.Marry(p2);
}
// instance method to marry
public void Marry(Person partner)
{
  if (married) return;
  spouse = partner;
  married = true;
  partner.Marry(this); // this is the current object
}
```

Note the following:

- In the `static` method named `Marry`, the `Person` objects are passed as parameters named `p1` and `p2` to the instance method `Marry`.
- In the instance method named `Marry`, the `spouse` is set to the `partner` passed as a parameter and the married Boolean is set to `true`.

1. In `PersonAutoGen.cs`, add one instance method and one static method to the `Person` class that will allow two `Person` objects to procreate, as shown in the following code:

```
// static method to "multiply"
public static Person Procreate(Person p1, Person p2)
{
  if (p1.Spouse != p2)
  {
    throw new ArgumentException("You must be married to procreate.");
  }
  Person baby = new()
  {
    Name = $"Baby of {p1.Name} and {p2.Name}",
    DateOfBirth = DateTime.Now
  };
  p1.Children.Add(baby);
  p2.Children.Add(baby);
  return baby;
}
// instance method to "multiply"
public Person ProcreateWith(Person partner)
{
  return Procreate(this, partner);
}
```

Note the following:

- In the `static` method named `Procreate`, the `Person` objects that will procreate are passed as parameters named `p1` and `p2`.
- A new `Person` class named `baby` is created with a name composed of a combination of the two people who have procreated. This could be changed later by setting the returned `baby` variable's `Name` property.
- The `baby` object is added to the `Children` collection of both parents and then returned. Classes are reference types, meaning a reference to the `baby` object stored in memory is added, not a clone of the `baby` object. You will learn the difference between reference types and value types later in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*.
- In the instance method named `ProcreateWith`, the `Person` object to procreate with is passed as a parameter named `partner`, and it, along with `this`, is passed to the static `Procreate` method to reuse the method implementation. `this` is a keyword that references the current instance of the class.

Good Practice: A method that creates a new object, or modifies an existing object, should return a reference to that object so that the caller can access the results.

We will tell the story of Lamech and his two wives and their children, as described at the following link: https://www.kingjamesbibleonline.org/Genesis-4-19/

1. In `Program.cs`, create three people and have them marry and then procreate with each other, noting that to add a double-quote character into a `string`, you must prefix it with a backslash character like this, `\"`, as shown in the following code:

```
Person lamech = new() { Name = "Lamech" };
Person adah = new() { Name = "Adah" };
Person zillah = new() { Name = "Zillah" };
lamech.Marry(adah);
Person.Marry(zillah, lamech);
WriteLine($"{lamech.Name} is married to {lamech.Spouse?.Name ?? "nobody"}");
WriteLine($"{adah.Name} is married to {adah.Spouse?.Name ?? "nobody"}");
WriteLine($"{zillah.Name} is married to {zillah.Spouse?.Name ?? "nobody"}");
// call instance method
Person baby1 = lamech.ProcreateWith(adah);
baby1.Name = "Jabal";
WriteLine($"{baby1.Name} was born on {baby1.DateOfBirth}");
// call static method
Person baby2 = Person.Procreate(zillah, lamech);
baby2.Name = "Tubalcain";
WriteLine($"{lamech.Name} has {lamech.Children.Count} children.");
WriteLine($"{adah.Name} has {adah.Children.Count} children.");
WriteLine($"{zillah.Name} has {zillah.Children.Count} children.");
for (int i = 0; i < lamech.Children.Count; i++)
{
  WriteLine(format: "{0}'s child #{1} is named \"{2}\".",
    arg0: lamech.Name, arg1: i, arg2: lamech[i].Name);
}
```

1. Run the code and view the result, as shown in the following output:

```
Lamech is married to Adah
Adah is married to Lamech
Zillah is married to Lamech
Jabal was born on 20/02/2022 16:26:35
Lamech has 2 children.
Adah has 1 children.
Zillah has 1 children.
Lamech's child #0 is named "Jabal".
Lamech's child #1 is named "Tubalcain".
```

Implementing functionality using operators

The `System.String` class has a `static` method named `Concat` that concatenates two string values and returns the result, as shown in the following code:

```
string s1 = "Hello ";
string s2 = "World!";
string s3 = string.Concat(s1, s2);
WriteLine(s3); // Hello World!
```

Calling a method like `Concat` works, but it might be more natural for a programmer to use the `+` symbol operator to "add" two `string` values together, as shown in the following code:

```
string s3 = s1 + s2;
```

A well-known biblical phrase is *Go forth and multiply*, meaning to procreate. Let's write code so that the `*` (multiply) symbol will allow two `Person` objects to procreate. And we will use the `+` operator to marry two people.

We do this by defining a `static` operator for the `*` symbol. The syntax is rather like a method, because in effect, an operator *is* a method, but uses a symbol instead of a method name, which makes the syntax more concise.

1. In `PersonAutoGen.cs`, create a `static` operator for the `+` symbol, as shown in the following code:

```
// operator to "marry"
public static bool operator +(Person p1, Person p2)
{
  Marry(p1, p2);
  return p1.Married && p2.Married; // confirm they are both now married
}
```

The return type for an operator does not need to match the types passed as parameters to the operator, but the return type cannot be `void`.

1. In `PersonAutoGen.cs`, create a `static` operator for the `*` symbol, as shown in the following code:

```
// operator to "multiply"
public static Person operator *(Person p1, Person p2)
{
  return Procreate(p1, p2);
}
```

Good Practice: Unlike methods, operators do not appear in IntelliSense lists for a type. For every operator that you define, make a method as well, because it may not be obvious to a programmer that the operator is available. The implementation of the operator can then call the method, reusing the code you have written. A second reason for providing a method is that operators are not supported by every language compiler; for example, although arithmetic operators like * are supported by Visual Basic and F#, there is no requirement that other languages support all operators supported by C#.

1. In `Program.cs`, comment out the statement that calls that static `Marry` method to marry Zillah and Lamech, and replace it with an `if` statement that uses the `+` operator to marry them, as shown in the following code:

```
// Person.Marry(zillah, lamech);
if (zillah + lamech)
{
  WriteLine($"{zillah.Name} and {lamech.Name} successfully got married.");
}
```

1. In `Program.cs`, after calling the `Procreate` method and before the statements that write the children to the console, use the `*` operator for Lamech to make two more babies with his wives Adah and Zillah, as shown highlighted in the following code:

```
// use operator to "multiply"
Person baby3 = lamech * adah;
baby3.Name = "Jubal";
Person baby4 = zillah * lamech;
baby4.Name = "Naamah";
```

1. Run the code and view the result, as shown in the following output:

```
Zillah and Lamech successfully got married.
Lamech is married to Adah
Adah is married to Lamech
Zillah is married to Lamech
Jabal was born on 20/02/2022 16:49:43
Lamech has 4 children.
Adah has 2 children.
Zillah has 2 children.
Lamech's child #0 is named "Jabal".
Lamech's child #1 is named "Tubalcain".
Lamech's child #2 is named "Jubal".
Lamech's child #3 is named "Naamah".
```

Implementing functionality using local functions

A language feature introduced in C# 7.0 is the ability to define a **local function**.

Local functions are the method equivalent of local variables. In other words, they are methods that are only accessible from within the containing method in which they have been defined. In other languages, they are sometimes called **nested** or **inner functions**.

Local functions can be defined anywhere inside a method: the top, the bottom, or even somewhere in the middle!

We will use a local function to implement a factorial calculation:

1. In `PersonAutoGen.cs`, add statements to define a `Factorial` function that uses a local function inside itself to calculate the result, as shown in the following code:

```
// method with a local function
public static int Factorial(int number)
{
  if (number < 0)
  {
    throw new ArgumentException(
      $"{nameof(number)} cannot be less than zero.");
  }
  return localFactorial(number);
  int localFactorial(int localNumber) // local function
  {
    if (localNumber == 0) return 1;
    return localNumber * localFactorial(localNumber - 1);
  }
}
```

1. In `Program.cs`, add a statement to call the `Factorial` function and write the return value to the console, as shown in the following code:

```
WriteLine($"5! is {Person.Factorial(5)}");
```

1. Run the code and view the result, as shown in the following output:

```
5! is 120
```

Pattern matching with objects

In *Chapter 3*, *Controlling Flow, Converting Types, and Handling Exceptions*, you were introduced to basic pattern matching. In this section, we will explore pattern matching in more detail.

Defining flight passengers

In this example, we will define some classes that represent various types of passengers on a flight and then we will use a switch expression with pattern matching to determine the cost of their flight.

1. In the `PacktLibrary` project/folder, add a new file named `FlightPatterns.cs`.
2. In `FlightPatterns.cs`, add statements to define three types of passenger with different properties, as shown in the following code:

```
namespace Packt.Shared;
public class Passenger
{
  public string? Name { get; set; }
}
public class BusinessClassPassenger : Passenger
{
  public override string ToString()
```

```
  {
    return $"Business Class: {Name}";
  }
}
public class FirstClassPassenger : Passenger
{
  public int AirMiles { get; set; }
  public override string ToString()
  {
    return $"First Class with {AirMiles:N0} air miles: {Name}";
  }
}
public class CoachClassPassenger : Passenger
{
  public double CarryOnKG { get; set; }
  public override string ToString()
  {
    return $"Coach Class with {CarryOnKG:N2} KG carry on: {Name}";
  }
}
```

You will learn about overriding the `ToString` method in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*.

1. In `Program.cs`, add statements to define an object array containing five passengers of various types and property values, and then enumerate them, outputting the cost of their flight, as shown in the following code:

```
Passenger[] passengers = {
  new FirstClassPassenger { AirMiles = 1_419, Name = "Suman" },
  new FirstClassPassenger { AirMiles = 16_562, Name = "Lucy" },
  new BusinessClassPassenger { Name = "Janice" },
  new CoachClassPassenger { CarryOnKG = 25.7, Name = "Dave" },
  new CoachClassPassenger { CarryOnKG = 0, Name = "Amit" },
};
foreach (Passenger passenger in passengers)
{
  decimal flightCost = passenger switch
  {
    FirstClassPassenger p when p.AirMiles > 35000 => 1500M,
    FirstClassPassenger p when p.AirMiles > 15000 => 1750M,
    FirstClassPassenger _                         => 2000M,
    BusinessClassPassenger _                      => 1000M,
    CoachClassPassenger p when p.CarryOnKG < 10.0 => 500M,
    CoachClassPassenger _                         => 650M,
    _                                             => 800M
  };
  WriteLine($"Flight costs {flightCost:C} for {passenger}");
}
```

While reviewing the preceding code, note the following:

- To pattern match on the properties of an object, you must name a local variable, like `p`, that can then be used in an expression.
- To pattern match on a type only, you can use `_` to discard the local variable.
- The switch expression also uses `_` to represent its default branch.

1. Run the code and view the result, as shown in the following output:

```
Flight costs £2,000.00 for First Class with 1,419 air miles
Flight costs £1,750.00 for First Class with 16,562 air miles
Flight costs £1,000.00 for Business Class
Flight costs £650.00 for Coach Class with 25.70 KG carry on
Flight costs £500.00 for Coach Class with 0.00 KG carry on
```

Enhancements to pattern matching in C# 9 or later

The previous examples worked with C# 8. Now we will look at some enhancements in C# 9 and later. First, you no longer need to use the underscore to discard when doing type matching:

1. In `Program.cs`, comment out the C# 8 syntax and add C# 9 and later syntax to modify the branches for first-class passengers to use a nested switch expression and the new support for conditionals like `>`, as shown in the following code:

```
decimal flightCost = passenger switch
{
  /* C# 8 syntax
  FirstClassPassenger p when p.AirMiles > 35000 => 1500M,
  FirstClassPassenger p when p.AirMiles > 15000 => 1750M,
  FirstClassPassenger                           => 2000M, */
  // C# 9 or later syntax
  FirstClassPassenger p => p.AirMiles switch
  {
    > 35000 => 1500M,
    > 15000 => 1750M,
    _       => 2000M
  },
  BusinessClassPassenger                        => 1000M,
  CoachClassPassenger p when p.CarryOnKG < 10.0 => 500M,
  CoachClassPassenger                           => 650M,
  _                                             => 800M
};
```

1. Run the code to view the results, and note they are the same as before.

You could also use the relational pattern in combination with the property pattern to avoid the nested switch expression, as shown in the following code:

```
FirstClassPassenger { AirMiles: > 35000 } => 1500,
FirstClassPassenger { AirMiles: > 15000 } => 1750M,
FirstClassPassenger                       => 2000M,
```

Working with records

Before we dive into the new records language feature, let us see some other related new features of C# 9 and later.

Init-only properties

You have used object initialization syntax to instantiate objects and set initial properties throughout this chapter. Those properties can also be changed after instantiation.

Sometimes you want to treat properties like `readonly` fields so they can be set during instantiation but not after. The new `init` keyword enables this. It can be used in place of the `set` keyword:

1. In the `PacktLibrary` project/folder, add a new file named `Records.cs`.
2. In `Records.cs`, define a person class with two immutable properties, as shown in the following code:

```
namespace Packt.Shared;
public class ImmutablePerson
{
  public string? FirstName { get; init; }
  public string? LastName { get; init; }
}
```

1. In `Program.cs`, add statements to instantiate a new immutable person and then try to change one of its properties, as shown in the following code:

```
ImmutablePerson jeff = new()
{
  FirstName = "Jeff",
```

```
  LastName = "Winger"
};
jeff.FirstName = "Geoff";
```

1. Compile the console app and note the compile error, as shown in the following output:

```
Program.cs(254,7): error CS8852: Init-only property or indexer 'ImmutablePerson.FirstName' can o
```

1. Comment out the attempt to set the `FirstName` property after instantiation.

Understanding records

Init-only properties provide some immutability to C#. You can take the concept further by using **records**. These are defined by using the `record` keyword instead of the `class` keyword. That can make the whole object immutable, and it acts like a value when compared. We will discuss equality and comparisons of classes, records, and value types in more detail in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*.

Records should not have any state (properties and fields) that changes after instantiation. Instead, the idea is that you create new records from existing ones. The new record has the changed state. This is called non-destructive mutation. To do this, C# 9 introduced the `with` keyword:

1. In `Records.cs`, add a record named `ImmutableVehicle`, as shown in the following code:

```
public record ImmutableVehicle
{
  public int Wheels { get; init; }
  public string? Color { get; init; }
  public string? Brand { get; init; }
}
```

1. In `Program.cs`, add statements to create a `car` and then a mutated copy of it, as shown in the following code:

```
ImmutableVehicle car = new()
{
  Brand = "Mazda MX-5 RF",
  Color = "Soul Red Crystal Metallic",
  Wheels = 4
};
ImmutableVehicle repaintedCar = car
  with { Color = "Polymetal Grey Metallic" };
WriteLine($"Original car color was {car.Color}.");
WriteLine($"New car color is {repaintedCar.Color}.");
```

1. Run the code to view the results, and note the change to the car color in the mutated copy, as shown in the following output:

```
Original car color was Soul Red Crystal Metallic.
New car color is Polymetal Grey Metallic.
```

Positional data members in records

The syntax for defining a record can be greatly simplified using positional data members.

Simplifying data members in records

Instead of using object initialization syntax with curly braces, sometimes you might prefer to provide a constructor with positional parameters, as you saw earlier in this chapter. You can also combine this with a deconstructor for splitting the object into individual parts, as shown in the following code:

```
public record ImmutableAnimal
{
```

```
  public string Name { get; init; }
  public string Species { get; init; }
  public ImmutableAnimal(string name, string species)
  {
    Name = name;
    Species = species;
  }
  public void Deconstruct(out string name, out string species)
  {
    name = Name;
    species = Species;
  }
}
```

The properties, constructor, and deconstructor can be generated for you:

1. In `Records.cs`, add statements to define another record using simplified syntax known as positional records, as shown in the following code:

```
// simpler way to define a record
// auto-generates the properties, constructor, and deconstructor
public record ImmutableAnimal(string Name, string Species);
```

1. In `Program.cs`, add statements to construct and deconstruct immutable animals, as shown in the following code:

```
ImmutableAnimal oscar = new("Oscar", "Labrador");
var (who, what) = oscar; // calls Deconstruct method
WriteLine($"{who} is a {what}.");
```

1. Run the application and view the results, as shown in the following output:

```
Oscar is a Labrador.
```

You will see records again when we look at C# 10 support for creating `struct` records in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*.

Practicing and exploring

Test your knowledge and understanding by answering some questions, getting some hands-on practice, and exploring this chapter's topics with deeper research.

Exercise 5.1 – Test your knowledge

Answer the following questions:

1. What are the six combinations of access modifier keywords and what do they do?
2. What is the difference between the `static`, `const`, and `readonly` keywords when applied to a type member?
3. What does a constructor do?
4. Why should you apply the `[Flags]` attribute to an `enum` type when you want to store combined values?
5. Why is the `partial` keyword useful?
6. What is a tuple?
7. What does the `record` keyword do?
8. What does overloading mean?
9. What is the difference between a field and a property?
10. How do you make a method parameter optional?

Exercise 5.2 – Explore topics

Use the links on the following page to learn more detail about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-5---building-your-own-types-with-object-oriented-programming

Summary

In this chapter, you learned about:

- Making your own types using OOP.
- Some of the different categories of members that a type can have, including fields to store data and methods to perform actions.
- OOP concepts, such as aggregation and encapsulation.
- How to use operators as an alternative to methods to implement simple functionality.
- How to use modern C# features like relational and property pattern matching enhancements, `init`-only properties, and records.

In the next chapter, you will take these concepts further by defining delegates and events, implementing interfaces, and inheriting from existing classes.

6 Implementing Interfaces and Inheriting Classes

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about deriving new types from existing ones using **object-oriented programming** (**OOP**). You will learn about generics and how they make your code safer and more performant. You will learn about delegates and events for exchanging messages between types. You will see the differences between reference and value types. You will implement interfaces for common functionality. You will create a derived class to inherit from a base class to reuse functionality, override an inherited type member, and use polymorphism. You will learn how to create extension methods and how to cast between classes in an inheritance hierarchy. Finally, you will learn how to write better code with the help of static code analyzers.

This chapter covers the following topics:

- Setting up a class library and console application
- Making types safely reusable with generics
- Raising and handling events
- Implementing interfaces
- Managing memory with reference and value types
- Working with null values
- Inheriting from classes
- Casting within inheritance hierarchies
- Inheriting and extending .NET types
- Using an analyzer to write better code

Setting up a class library and console application

We will start by defining a workspace/solution with two projects like the one created in *Chapter 5*, *Building Your Own Types with Object-Oriented Programming*. Even if you completed all the exercises in that chapter, follow the instructions below so that you start this chapter with fresh working projects:

1. Use your preferred coding tool to create a new project, as defined in the following list:
    - Project template: **Class Library** / `classlib`
    - Project file and folder: `PacktLibrary`
    - Workspace/solution file and folder: `Chapter06`
2. Add a new project, as defined in the following list:
    - Project template: **Console App** / `console`
    - Project file and folder: `PeopleApp`
    - Workspace/solution file and folder: `Chapter06`
3. In the `PacktLibrary` project, rename the file named `Class1.cs` to `Person.cs`.
4. In both projects, add an `<ItemGroup>` to globally and statically import the `System.Console` class, as shown in the following markup:

```
<ItemGroup>
    <Using Include="System.Console" Static="true" />
</ItemGroup>
```

1. Modify the `Person.cs` file contents, as shown in the following code:

```
namespace Packt.Shared;
public class Person : object
{
  // properties
  public string? Name { get; set; }
  public DateTime DateOfBirth { get; set; }
  // methods
  public void WriteToConsole()
  {
    WriteLine($"{Name} was born on a {DateOfBirth:dddd}.");
  }
}
```

1. In the `PeopleApp` project, add a project reference to `PacktLibrary`, as shown in the following markup:

```
<ItemGroup>
  <ProjectReference
    Include="..\PacktLibrary\PacktLibrary.csproj" />
</ItemGroup>
```

1. In `Program.cs`, delete the existing statements and then write statements to create an instance of `Person` and write information about it to the console, as shown in the following code:

```
using Packt.Shared;
Person harry = new()
{
  Name = "Harry",
  DateOfBirth = new(year: 2001, month: 3, day: 25)
};
harry.WriteToConsole();
```

1. Run the `PeopleApp` project and note the result, as shown in the following output:

```
Harry was born on a Sunday.
```

Making types safely reusable with generics

In 2005, with C# 2.0 and .NET Framework 2.0, Microsoft introduced a feature named **generics**, which enables your types to be more safely reusable and more efficient. It does this by allowing a programmer to pass types as parameters, like how you can pass objects as parameters.

Working with non-generic types

First, let's look at an example of working with a non-generic type so that you can understand the problem that generics are designed to solve, such as weakly typed parameters and values, and performance problems caused by using `System.Object`.

`System.Collections.Hashtable` can be used to store multiple values each with a unique key that can later be used to quickly look up its value. Both the key and value can be any object because they are declared as `System.Object`. Although this provides flexibility when storing value types like integers, it is slow, and bugs are easier to introduce because no type checks are made when adding items.

Let's write some code:

1. In `Program.cs`, create an instance of the non-generic collection, `System.Collections.Hashtable`, and then add four items to it, as shown in the following code:

```
// non-generic lookup collection
System.Collections.Hashtable lookupObject = new();
lookupObject.Add(key: 1, value: "Alpha");
lookupObject.Add(key: 2, value: "Beta");
```

```
lookupObject.Add(key: 3, value: "Gamma");
lookupObject.Add(key: harry, value: "Delta");
```

1. Add statements to define a `key` with the value of `2` and use it to look up its value in the hash table, as shown in the following code:

```
int key = 2; // look up the value that has 2 as its key
WriteLine(format: "Key {0} has value: {1}",
  arg0: key,
  arg1: lookupObject[key]);
```

1. Add statements to use the `harry` object to look up its value, as shown in the following code:

```
// look up the value that has harry as its key
WriteLine(format: "Key {0} has value: {1}",
  arg0: harry,
  arg1: lookupObject[harry]);
```

1. Run the code and note that it works, as shown in the following output:

```
Key 2 has value: Beta
Key Packt.Shared.Person has value: Delta
```

Although the code works, there is potential for mistakes because literally any type can be used for the key or value. If another developer used your lookup object and expected all the items to be a certain type, they might cast them to that type and get exceptions because some values might be a different type. A lookup object with lots of items would also give poor performance.

Good Practice: Avoid types imported in the `System.Collections` namespace. Use types imported in `System.Collections.Generics` and other namespaces instead.

Working with generic types

`System.Collections.Generic.Dictionary<TKey, TValue>` can be used to store multiple values each with a unique key that can later be used to quickly look up its value. Both the key and value can be any object, but you must tell the compiler what the types of the key and value will be when you first instantiate the collection. You do this by specifying types for the **generic parameters** in angle brackets `<>`, `TKey`, and `TValue`.

Good Practice: When a generic type has one definable type, it should be named `T`, for example, `List<T>`, where `T` is the type stored in the list. When a generic type has multiple definable types, they should use `T` as a name prefix and have a sensible name, for example, `Dictionary<TKey, TValue>`.

This provides flexibility, it is faster, and bugs are easier to avoid because type checks are made when adding items. We will not need to explicitly specify the `System.Collections.Generic` namespace that contains `Dictionary<TKey, TValue>` because it is implicitly and globally imported by default.

Let's write some code to solve the problem by using generics:

1. In `Program.cs`, create an instance of the generic lookup collection `Dictionary<TKey, TValue>` and then add four items to it, as shown in the following code:

```
// generic lookup collection
Dictionary<int, string> lookupIntString = new();
lookupIntString.Add(key: 1, value: "Alpha");
lookupIntString.Add(key: 2, value: "Beta");
lookupIntString.Add(key: 3, value: "Gamma");
lookupIntString.Add(key: harry, value: "Delta");
```

1. Note the compile error when using `harry` as a key, as shown in the following output:

```
/Users/markjprice/Code/Chapter06/PeopleApp/Program.cs(98,32): error CS1503: Argument 1: cannot c
```

1. Replace `harry` with `4`.
2. Add statements to set the `key` to `3` and use it to look up its value in the dictionary, as shown in the following code:

```
key = 3;
WriteLine(format: "Key {0} has value: {1}",
  arg0: key,
  arg1: lookupIntString[key]);
```

1. Run the code and note that it works, as shown in the following output:

```
Key 3 has value: Gamma
```

Raising and handling events

Methods are often described as *actions that an object can perform, either on itself or on related objects*. For example, `List<T>` can add an item to itself or clear itself, and `File` can create or delete a file in the filesystem.

Events are often described as *actions that happen to an object*. For example, in a user interface, `Button` has a `Click` event, a click being something that happens to a button, and `FileSystemWatcher` listens to the filesystem for change notifications and raises events like `Created` and `Deleted` that are triggered when a directory or file changes.

Another way of thinking of events is that they provide a way of exchanging messages between two objects.

Events are built on **delegates**, so let's start by having a look at what delegates are and how they work.

Calling methods using delegates

You have already seen the most common way to call or execute a method: using the `.` operator to access the method using its name. For example, `Console.WriteLine` tells the `Console` type to call its `WriteLine` method.

The other way to call or execute a method is to use a delegate. If you have used languages that support **function pointers**, then think of a delegate as being a **type-safe method pointer**.

In other words, a delegate contains the memory address of a method that matches the same signature as the delegate so that it can be called safely with the correct parameter types.

For example, imagine there is a method in the `Person` class that must have a `string` type passed as its only parameter, and it returns an `int` type, as shown in the following code:

```
public int MethodIWantToCall(string input)
{
  return input.Length; // it doesn't matter what the method does
}
```

I can call this method on an instance of `Person` named `p1` like this:

```
Person p1 = new();
int answer = p1.MethodIWantToCall("Frog");
```

Alternatively, I can define a delegate with a matching signature to call the method indirectly. Note that the names of the parameters do not have to match. Only the types of parameters and return values must match, as shown in the following code:

```
delegate int DelegateWithMatchingSignature(string s);
```

Now, I can create an instance of the delegate, point it at the method, and finally call the delegate (which calls the method), as shown in the following code:

```
// create a delegate instance that points to the method
DelegateWithMatchingSignature d = new(p1.MethodIWantToCall);
// call the delegate, who then calls the method
int answer2 = d("Frog");
```

You are probably thinking, "What's the point of that?" Well, it provides flexibility.

For example, we could use delegates to create a queue of methods that need to be called in order. Queuing actions that need to be performed is common in services to provide improved scalability.

Another example is to allow multiple actions to perform in parallel. Delegates have built-in support for asynchronous operations that run on a different thread, and that can provide improved responsiveness.

The most important example is that delegates allow us to implement events for sending messages between different objects that do not need to know about each other. Events are an example of loose coupling between components because the components do not need to know about each other; they just need to know the event signature.

Delegates and events are two of the most confusing features of C# and can take a few attempts to understand, so don't worry if you feel lost as we walk through how they work!

Defining and handling delegates

Microsoft has two predefined delegates for use as events. They both have two parameters:

- `object? sender`: This parameter is a reference to the object raising the event or sending the message. The reference could be `null`.
- `EventArgs e` or `TEventArgs e`: This parameter contains additional relevant information about the event that is occurring. For example, in a GUI app, you might define `MouseMoveEventArgs` that has properties for the `X` and `Y` co-ordinates for the mouse pointer. A bank account might have a `WithdrawEventArgs` with a property for the `amount` to withdraw.

Their signatures are simple, yet flexible, as shown in the following code:

```
// for methods that do not need additional argument values passed in
public delegate void EventHandler(object? sender, EventArgs e);
// for methods that need additional argument values passed in as
// defined by the generic type TEventArgs
public delegate void EventHandler<TEventArgs>(object? sender, TEventArgs e);
```

> **Good Practice**: When you want to define an event in your own types, you should use one of these two predefined delegates.

Let's explore delegates and events:

1. Add statements to the `Person` class and note the following points, as shown in the following code:
   - It defines an `EventHandler` delegate field named `Shout`.
   - It defines an `int` field to store `AngerLevel`.
   - It defines a method named `Poke`.
   - Each time a person is poked, their `AngerLevel` increments. Once their `AngerLevel` reaches three, they raise the `Shout` event, but only if there is at least one event delegate pointing at a method defined somewhere else in the code; that is, it is not `null`:

```
// delegate field
public EventHandler? Shout;
// data field
public int AngerLevel;
// method
public void Poke()
{
  AngerLevel++;
```

```
  if (AngerLevel >= 3)
  {
    // if something is listening...
    if (Shout != null)
    {
      // ...then call the delegate
      Shout(this, EventArgs.Empty);
    }
  }
}
```

Checking whether an object is not `null` before calling one of its methods is very common. C# 6.0 and later allows `null` checks to be simplified inline using a `?` symbol before the `.` operator, as shown in the following code:

```
Shout?.Invoke(this, EventArgs.Empty);
```

1. In the `PeopleApp` project, add a new class file named `Program.EventHandlers.cs`.
2. In `Program.EventHandlers.cs`, add a method with a matching signature that gets a reference to the `Person` object from the `sender` parameter and outputs some information about them, as shown in the following code:

```
using Packt.Shared;
partial class Program
{
  // a method to handle the Shout event received by the harry object
  static void Harry_Shout(object? sender, EventArgs e)
  {
    if (sender is null) return;
    Person? p = sender as Person;
    if (p is null) return;
    WriteLine($"{p.Name} is this angry: {p.AngerLevel}.");
  }
}
```

Microsoft's convention for method names that handle events is `ObjectName_EventName`.

1. In `Program.cs`, add a statement to assign the method to the delegate field, and then add statements to call the `Poke` method four times, as shown in the following code:

```
// assign a method to the Shout delegate
harry.Shout = Harry_Shout;
// call the Poke method that raises the Shout event
harry.Poke();
harry.Poke();
harry.Poke();
harry.Poke();
```

1. Run the code and view the result, and note that Harry says nothing the first two times he is poked, and only gets angry enough to shout once he's been poked at least three times, as shown in the following output:

```
Harry is this angry: 3.
Harry is this angry: 4.
```

Defining and handling events

You've now seen how delegates implement the most important functionality of events: the ability to define a signature for a method that can be implemented by a completely different piece of code, and then call that method and any others that are hooked up to the delegate field.

But what about events? There is less to them than you might think.

When assigning a method to a delegate field, you should not use the simple assignment operator as we did in the preceding example.

Delegates are multicast, meaning that you can assign multiple delegates to a single delegate field. Instead of the `=` assignment, we could have used the `+=` operator so we could add more methods to the same delegate field. When the delegate is called, all the assigned methods are called, although you have no control over the order in which they are called.

If the `Shout` delegate field was already referencing one or more methods, by assigning a method, it would replace all the others. With delegates that are used for events, we usually want to make sure that a programmer only ever uses either the `+=` operator or the `-=` operator to assign and remove methods:

1. To enforce this, in `Person.cs`, add the `event` keyword to the delegate field declaration, as shown highlighted in the following code:

```
public event EventHandler? Shout;
```

1. Build the `PeopleApp` project and note the compiler error message, as shown in the following output:

```
Program.cs(41,13): error CS0079: The event 'Person.Shout' can only appear on the left hand side o
```

This is (almost) all that the `event` keyword does! If you will never have more than one method assigned to a delegate field, then technically you do not need "events," but it is still good practice to indicate your meaning and that you expect a delegate field to be used as an event.

1. In `Program.cs`, modify the comment and the method assignment to use `+=`, as shown in the following code:

```
// assign event handler methods to Shout event
harry.Shout += Harry_Shout;
```

1. Run the code and note that it has the same behavior as before.
2. In `Program.EventHandlers.cs`, copy and paste the method, and then change its comment, name, and output, as shown highlighted in the following code:

```
// another method to handle the Shout event received by the harry object
static void Harry_Shout2(object? sender, EventArgs e)
{
  if (sender is null) return;
  Person? p = sender as Person;
  if (p is null) return;
  WriteLine($"Stop it!");
}
```

1. In `Program.cs`, after the statement that assigns the `Harry_Shout` method to the `Shout` event, add a statement to attach the new event handler to the `Shout` event too, as shown highlighted in the following code:

```
harry.Shout += Harry_Shout;
harry.Shout += Harry_Shout2;
```

1. Run the code and view the result, and note that both event handlers execute whenever the event is raised, as shown in the following output:

```
Harry is this angry: 3.
Stop it!
Harry is this angry: 4.
Stop it!
```

Implementing interfaces

Interfaces are a way to implement standard functionality and of connecting different types to make new things. Think of them like the studs on top of LEGO™ bricks, which allow them to "stick" together, or electrical standards for plugs and sockets.

If a type implements an interface, then it is making a promise to the rest of .NET that it supports specific functionality. Therefore, they are sometimes described as contracts.

Common interfaces

Here are some common interfaces that your types might implement:

Interface	Method(s)	Description
`IComparable`	`CompareTo(other)`	This defines a comparison method that a type implements to order or sort its instances.
`IComparer`	`Compare(first, second)`	This defines a comparison method that a secondary type implements to order or sort instances of a primary type.
`IDisposable`	`Dispose()`	This defines a disposal method to release unmanaged resources more efficiently than waiting for a finalizer. See the *Releasing unmanaged resources* section later in this chapter for more details.
`IFormattable`	`ToString(format, culture)`	This defines a culture-aware method to format the value of an object into a string representation.
`IFormatter`	`Serialize(stream, object)` `Deserialize(stream)`	This defines methods to convert an object to and from a stream of bytes for storage or transfer.
`IFormatProvider`	`GetFormat(type)`	This defines a method to format inputs based on a language and region.

Comparing objects when sorting

One of the most common interfaces that you will want to implement is `IComparable`. It has one method named `CompareTo`. It has two variations, one that works with a nullable `object` type and one that works with a nullable generic type `T`, as shown in the following code:

```
namespace System
{
  public interface IComparable
  {
    int CompareTo(object? obj);
  }
  public interface IComparable<in T>
  {
    int CompareTo(T? other);
  }
}
```

For example, the `string` type implements `IComparable` by returning `-1` if the `string` should be sorted before the `string` being compared to, `1` if it should be sorted after, and `0` if they are equal. The `int` type implements `IComparable` by returning `-1` if the `int` is less than the `int` being compared to, `1` if it is greater, and `0` if they are equal.

If a type implements one of the `IComparable` interfaces, then arrays and collections containing instances of that type can be sorted.

Before we implement the `IComparable` interface and its `CompareTo` method for the `Person` class, let's see what happens when we try to sort an array of `Person` instances, including some that are `null` or have a `null` value for their `Name` property:

1. In the `PeopleApp` project, add a new class file named `Program.Helpers.cs`.
2. In `Program.Helpers.cs`, add a method to the `Program` class that will output all the names of a collection of people passed as a parameter with a title beforehand, as shown in the following code:

```
using Packt.Shared;
partial class Program
{
  static void OutputPeopleNames(IEnumerable<Person?> people, string title)
  {
    WriteLine(title);
    foreach (Person? p in people)
    {
      WriteLine("  {0}",
        p is null ? "<null> Person" : p.Name ?? "<null> Name");
      /* if p is null then output: <null> Person
         else output: p.Name
         unless p.Name is null in which case output: <null> Name */
    }
  }
}
```

1. In `Program.cs`, add statements that create an array of `Person` instances and call the `OutputPeopleNames` method to write the items to the console, and then attempt to sort the array and write the items to the console again, as shown in the following code:

```
Person?[] people =
{
  null,
  new() { Name = "Simon" },
  new() { Name = "Jenny" },
  new() { Name = "Adam" },
  new() { Name = null },
  new() { Name = "Richard" }
};
OutputPeopleNames(people, "Initial list of people:");
Array.Sort(people);
OutputPeopleNames(people,
  "After sorting using Person's IComparable implementation:");
```

1. Run the code and an exception will be thrown. As the message explains, to fix the problem, our type must implement `IComparable`, as shown in the following output:

```
Unhandled Exception: System.InvalidOperationException: Failed to compare two elements in the arra
```

1. In `Person.cs`, after inheriting from `object`, add a comma and enter `IComparable<Person?>`, as shown highlighted in the following code:

```
public class Person : object, IComparable<Person?>
```

Your code editor will draw a red squiggle under the new code to warn you that you have not yet implemented the method you have promised to. Your code editor can write the skeleton implementation for you.

1. Click on the light bulb and then click **Implement interface**.
2. Scroll down to the bottom of the `Person` class to find the method that was written for you, as shown in the following code:

```
public int CompareTo(Person? other)
{
  throw new NotImplementedException();
}
```

1. Delete the statement that throws the `NotImplementedException` error.
2. Add statements to handle variations of input values, including `null`, and call the `CompareTo` method of the `Name` field, which uses the `string` type's implementation of `CompareTo`, and return the result, as shown in the following code:

```
int position;
if ((this is not null) && (other is not null))
{
  if ((Name is not null) && (other.Name is not null))
  {
    // if both Name values are not null,
    // use the string implementation of CompareTo
    position = Name.CompareTo(other.Name);
  }
  else if ((Name is not null) && (other.Name is null))
  {
    position = -1; // else this Person precedes other Person
  }
  else if ((Name is null) && (other.Name is not null))
  {
    position = 1; // else this Person follows other Person
  }
  else
  {
    position = 0; // this Person and other Person are at same position
  }
}
else if ((this is not null) && (other is null))
{
  position = -1; // this Person precedes other Person
}
else if ((this is null) && (other is not null))
{
  position = 1; // this Person follows other Person
}
else
{
  position = 0; // this Person and other Person are at same position
}
return position;
```

We have chosen to compare two `Person` instances by comparing their `Name` fields. `Person` instances will, therefore, be sorted alphabetically by their name. `null` values will be sorted to the bottom of the collection. Storing the calculated `position` before returning it is useful when debugging.

1. Run the `PeopleApp` console app and note that this time it works as it should, sorted alphabetically by name, as shown in the following output:

```
Initial list of people:
  Simon
  <null> Person
  Jenny
  Adam
  <null> Name
  Richard
After sorting using Person's IComparable implementation:
  Adam
  Jenny
  Richard
  Simon
  <null> Name
  <null> Person
```

Good Practice: If anyone will want to sort an array or collection of instances of your type, then implement the `IComparable` interface.

Comparing objects using a separate class

Sometimes, you won't have access to the source code for a type, and it might not implement the `IComparable` interface. Luckily, there is another way to sort instances of a type. You can create a separate type that implements a slightly different interface, named `IComparer`:

1. In the `PacktLibrary` project, add a new class file named `PersonComparer.cs` containing a class implementing the `IComparer` interface that will compare two people, that is, two `Person` instances. Implement it by comparing the length of their `Name` field, or if the names are the same length, then by comparing the names alphabetically, as shown in the following code:

```
namespace Packt.Shared;
public class PersonComparer : IComparer<Person?>
{
  public int Compare(Person? x, Person? y)
  {
    int position;
    if ((x is not null) && (y is not null))
    {
      if ((x.Name is not null) && (y.Name is not null))
      {
        // if both Name values are not null...
        // ...compare the Name lengths...
        int result = x.Name.Length.CompareTo(y.Name.Length);
        /// ...if they are equal...
        if (result == 0)
        {
          // ...then compare by the Names...
          return x.Name.CompareTo(y.Name);
        }
        else
        {
          // ...otherwise compare by the lengths.
          position = result;
        }
      }
      else if ((x.Name is not null) && (y.Name is null))
      {
        position = -1; // else x Person precedes y Person
      }
      else if ((x.Name is null) && (y.Name is not null))
      {
        position = 1; // else x Person follows y Person
      }
      else
      {
        position = 0; // x Person and y Person are at same position
      }
    }
    else if ((x is not null) && (y is null))
    {
      position = -1; // x Person precedes y Person
    }
    else if ((x is null) && (y is not null))
    {
      position = 1; // x Person follows y Person
    }
    else
    {
      position = 0; // x Person and y Person are at same position
    }
    return position;
  }
}
```

1. In `Program.cs`, add statements to sort the array using this alternative implementation, as shown in the following code:

```
Array.Sort(people, new PersonComparer());
OutputPeopleNames(people,
  "After sorting using PersonComparer's IComparer implementation:");
```

1. Run the `PeopleApp` console app and view the result of sorting the people by length of name and then alphabetically, as shown in the following output:

```
After sorting using PersonComparer's IComparer implementation:
  Adam
  Jenny
  Simon
  Richard
  <null> Name
  <null> Person
```

This time, when we sort the `people` array, we explicitly ask the sorting algorithm to use the `PersonComparer` type instead, so that the people are sorted with the shortest names first, like `Adam`, and the longest names last, like `Richard`; and when the lengths of two or more names are equal, to sort them alphabetically, like `Jenny` and `Simon`.

Implicit and explicit interface implementations

Interfaces can be implemented implicitly and explicitly. Implicit implementations are simpler and more common. Explicit implementations are only necessary if a type must have multiple methods with the same name and signature.

For example, both `IGamePlayer` and `IKeyHolder` might have a method called `Lose` with the same parameters because both a game and a key can be lost. In a type that must implement both interfaces, only one implementation of `Lose` can be the implicit method. If both interfaces can share the same implementation, that works, but if not then the other `Lose` method will have to be implemented differently and called explicitly, as shown in the following code:

```
public interface IGamePlayer
{
  void Lose();
}
public interface IKeyHolder
{
  void Lose();
}
public class Person : IGamePlayer, IKeyHolder
{
  public void Lose() // implicit implementation
  {
    // implement losing a key
  }
  void IGamePlayer.Lose() // explicit implementation
  {
    // implement losing a game
  }
}
// calling implicit and explicit implementations of Lose
Person p = new();
p.Lose(); // calls implicit implementation of losing a key
((IGamePlayer)p).Lose(); // calls explicit implementation of losing a game
IGamePlayer player = p as IGamePlayer;
player.Lose(); // calls explicit implementation of losing a game
```

Defining interfaces with default implementations

A language feature introduced in C# 8.0 is **default implementations** for an interface. Let's see it in action:

1. In the `PacktLibrary` project, add a new file named `IPlayable.cs`.
2. Modify the statements to define a public `IPlayable` interface with two methods to `Play` and `Pause`, as shown in the following code:

```
namespace Packt.Shared;
public interface IPlayable
```

```
{
  void Play();
  void Pause();
}
```

1. In the `PacktLibrary` project, add a new class file named `DvdPlayer.cs`.
2. Modify the statements in the file to implement the `IPlayable` interface, as shown in the following code:

```
namespace Packt.Shared;
public class DvdPlayer : IPlayable
{
  public void Pause()
  {
    WriteLine("DVD player is pausing.");
  }
  public void Play()
  {
    WriteLine("DVD player is playing.");
  }
}
```

This is useful, but what if we decide to add a third method named `Stop`? Before C# 8.0, this would be impossible once at least one type implemented the original interface. One of the main points of an interface is that it is a fixed contract.

C# 8.0 allows an interface to add new members after release as long as they have a default implementation. C# purists do not like the idea, but for practical reasons, such as avoiding breaking changes or having to define a whole new interface, it is useful, and other languages such as Java and Swift enable similar techniques.

Support for default interface implementations requires some fundamental changes to the underlying platform, so they are only supported with C# if the target framework is .NET 5.0 or later, .NET Core 3.0 or later, or .NET Standard 2.1. They are therefore not supported by .NET Framework.

1. Modify the `IPlayable` interface to add a `Stop` method with a default implementation, as shown highlighted in the following code:

```
namespace Packt.Shared;
public interface IPlayable
{
  void Play();
  void Pause();
  void Stop() // default interface implementation
  {
    WriteLine("Default implementation of Stop.");
  }
}
```

1. Build the `PeopleApp` project and note that the projects compile successfully despite the `DvdPlayer` class not implementing `Stop`. In the future, we could override the default implementation of `Stop` by implementing it in the `DvdPlayer` class.

Managing memory with reference and value types

I have mentioned **reference types** a couple of times. Let's look at them in more detail.

There are two categories of memory: **stack** memory and **heap** memory. With modern operating systems, the stack and heap can be anywhere in physical or virtual memory.

Stack memory is faster to work with (because it is managed directly by the CPU and because it uses a last-in, first-out mechanism, it is more likely to have the data in its L1 or L2 cache) but limited in size, while heap memory is slower but much more plentiful.

On Windows, for ARM64, x86, and x64 machines, the default stack size is 1 MB. It is 8 MB on a typical modern Linux-based operating system. For example, in a macOS terminal, I can enter the command `ulimit -a` to discover that the stack size is limited to 8,192 KB and that other memory is "unlimited." This limited amount of stack memory is why it is so easy to fill it up and get a "stack overflow."

Defining reference and value types

There are three C# keywords that you can use to define object types: `class`, `record`, and `struct`. All can have the same members, such as fields and methods. One difference between them is how memory is allocated:

- When you define a type using `record` or `class`, you are defining a **reference type**. This means that the memory for the object itself is allocated on the heap, and only the memory address of the object (and a little overhead) is stored on the stack.
- When you define a type using `record struct` or `struct`, you are defining a **value type**. This means that the memory for the object itself is allocated on the stack.

If a `struct` uses field types that are not of the `struct` type, then those fields will be stored on the heap, meaning the data for that object is stored in both the stack and the heap!

These are the most common struct types:

- **Number** `System` **types**: `byte`, `sbyte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`, `float`, `double`, and `decimal`.
- **Other** `System` **types**: `char`, `DateTime`, `DateOnly`, `TimeOnly`, and `bool`.
- `System.Drawing` **types**: `Color`, `Point`, `PointF`, `Size`, `SizeF`, `Rectangle`, and `RectangleF`.

Almost all the other types are `class` types, including `string` aka `System.String` and `object` aka `System.Object`.

> Apart from the difference in terms of where in memory the data for a type is stored, the other major difference is that you cannot inherit from a `struct`.

How reference and value types are stored in memory

Imagine that you have a console app that declares some variables, as shown in the following code:

```
int number1 = 49;
long number2 = 12;
System.Drawing.Point location = new(x: 4, y: 5);
Person kevin = new() { Name = "Kevin",
  DateOfBirth = new(year: 1988, month: 9, day: 23) };
Person sally;
```

Let's review what memory is allocated on the stack and heap when these statements execute, as shown in *Figure 6.1* and as described in the following list:

- The `number1` variable is a value type (also known as `struct`) so it is allocated on the stack, and it uses 4 bytes of memory since it is a 32-bit integer. Its value, `49`, is stored directly in the variable.
- The `number2` variable is also a value type so it is also allocated on the stack, and it uses 8 bytes since it is a 64-bit integer.
- The `location` variable is also a value type, so it is allocated on the stack, and it uses 8 bytes since it is made up of two 32-bit integers, `x` and `y`.
- The `kevin` variable is a reference type (also known as `class`) so 8 bytes for a 64-bit memory address (assuming a 64-bit operating system) is allocated on the stack and enough bytes on the heap to store an instance of a `Person`.
- The `sally` variable is a reference type, so 8 bytes for a 64-bit memory address is allocated on the stack. It is currently `null`, meaning no memory has yet been allocated for it on the heap. If we were to later assign

`kevin` to `sally` then the memory address of the `Person` on the heap would be copied into `sally`, as shown in the following code:

```
sally = kevin; // both variables point at the same Person on heap
```

Figure 6.1: How value and reference types are allocated in the stack and heap

All the allocated memory for a reference type is stored on the heap. If a value type such as `DateTime` is used for a field of a reference type like `Person`, then the `DateTime` value is stored on the heap, as shown in *Figure 6.1*.

If a value type has a field that is a reference type, then that part of the value type is stored on the heap. `Point` is a value type that consists of two fields, both of which are themselves value types, so the entire object can be allocated on the stack. If the `Point` value type had a field that was a reference type, like `string`, then the `string` bytes would be stored on the heap.

Equality of types

It is common to compare two variables using the `==` and `!=` operators. The behavior of these two operators is different for reference types and value types.

When you check the equality of two value type variables, .NET literally compares the *values* of those two variables on the stack and returns `true` if they are equal.

1. In `Program.cs`, add statements to declare two integers with equal values and then compare them, as shown in the following code:

```
int a = 3;
int b = 3;
WriteLine($"a: {a}, b: {b}");
WriteLine($"a == b: {(a == b)}");
```

1. Run the console app and view the result, as shown in the following output:

```
a: 3, b: 3
a == b: True
```

When you check the equality of two reference type variables, .NET compares the memory addresses of those two variables and returns `true` if they are equal.

1. In `Program.cs`, add statements to declare two `Person` instances with equal names and then compare the variables and their names, as shown in the following code:

```
Person p1 = new() { Name = "Kevin" };
Person p2 = new() { Name = "Kevin" };
WriteLine($"p1: {p1}, p2: {p2}");
WriteLine($"p1 == p2: {(p1 == p2)}");
```

1. Run the console app and view the result, as shown in the following output:

```
p1: Packt.Shared.Person, p2: Packt.Shared.Person
p1 == p2: False
```

This is because they are not the same object. If both variables literally pointed to the same object on the heap, then they would be equal.

1. Add statements to declare a third `Person` object and assign `p1` to it, as shown in the following code:

```
Person p3 = p1;
WriteLine($"p3: {p3}");
WriteLine($"p1 == p3: {(p1 == p3)}");
```

1. Run the console app and view the result, as shown in the following output:

```
p3: Packt.Shared.Person
p1 == p3: True
```

The one exception to this behavior of reference types is the `string` type. It is a reference type, but the equality operators have been overridden to make them behave as if they were value types.

1. Add statements to compare the `Name` properties of two `Person` instances, as shown in the following code:

```
WriteLine($"p1.Name: {p1.Name}, p2.Name: {p2.Name}");
WriteLine($"p1.Name == p2.Name: {(p1.Name == p2.Name)}");
```

1. Run the console app and view the result, as shown in the following output:

```
p1.Name: Kevin, p2.Name: Kevin
p1.Name == p2.Name: True
```

You can do something similar with your classes to make the equality operators return `true` even if they are not the same object (same memory address on the heap) but instead if their fields have the same values; that is beyond the scope of this book. Alternatively, use a `record class`, because one of their benefits is that they implement this behavior for you.

Defining struct types

Let's explore defining your own value types:

1. In the `PacktLibrary` project, add a file named `DisplacementVector.cs`.
2. Modify the file, as shown in the following code, and note the following:
   - The type is declared using `struct` instead of `class`.
   - It has two `int` properties, named `X` and `Y`, that will auto-generate two private fields with the same data type that will be allocated on the stack.
   - It has a constructor for setting initial values for `X` and `Y`.
   - It has an operator for adding two instances together that returns a new instance of the type with `X` added to `X`, and `Y` added to `Y`.

```
namespace Packt.Shared;
public struct DisplacementVector
{
  public int X { get; set; }
  public int Y { get; set; }
  public DisplacementVector(int initialX, int initialY)
  {
    X = initialX;
    Y = initialY;
  }
  public static DisplacementVector operator +(
    DisplacementVector vector1,
    DisplacementVector vector2)
  {
    return new(
      vector1.X + vector2.X,
      vector1.Y + vector2.Y);
  }
}
```

1. In `Program.cs`, add statements to create two new instances of `DisplacementVector`, add them together, and output the result, as shown in the following code:

```
DisplacementVector dv1 = new(3, 5);
DisplacementVector dv2 = new(-2, 7);
DisplacementVector dv3 = dv1 + dv2;
WriteLine($"({dv1.X}, {dv1.Y}) + ({dv2.X}, {dv2.Y}) = ({dv3.X}, {dv3.Y})");
```

1. Run the code and view the result, as shown in the following output:

```
(3, 5) + (-2, 7) = (1, 12)
```

Value types always have a default constructor even if an explicit one is not defined because the values on the stack must be initialized, even if to default values. For the two integer fields in `DisplacementVector`, they will be initialized to `0`.

1. In `Program.cs`, add statements to create a new instance of `DisplacementVector`, and output the object's properties, as shown in the following code:

```
DisplacementVector dv4 = new();
WriteLine($"({dv4.X}, {dv4.Y})");
```

1. Run the code and view the result, as shown in the following output:

```
(0, 0)
```

Good Practice: If the total memory used by all the fields in your type is 16 bytes or less, your type only uses value types for its fields, and you will never want to derive from your type, then Microsoft recommends that you use `struct`. If your type uses more than 16 bytes of stack memory, if it uses reference types for its fields, or if you might want to inherit from it, then use `class`.

Defining record struct types

C# 10 introduced the ability to use the `record` keyword with `struct` types as well as with `class` types.

We could define the `DisplacementVector` type, as shown in the following code:

```
public record struct DisplacementVector(int X, int Y);
```

A `record struct` has all the same benefits over a `record class` that a `struct` has over a `class`. One difference between `record struct` and `record class` is that `record struct` is not immutable unless you also apply the `readonly` keyword to the `record struct` declaration. A `struct` does not implement the `==` and `!=` operators but they are automatically implemented with a `record struct`.

With this change, Microsoft recommends explicitly specifying `class` if you want to define a `record class` even though the `class` keyword is optional, as shown in the following code:

```
public record class ImmutableAnimal(string Name);
```

Releasing unmanaged resources

In the previous chapter, we saw that constructors can be used to initialize fields and that a type may have multiple constructors. Imagine that a constructor allocates an **unmanaged resource**; that is, anything that is not controlled by .NET, such as a file or mutex under the control of the operating system. The unmanaged resource must be manually released because .NET cannot do it for us using its automatic garbage collection feature.

Garbage collection is an advanced topic, so for this topic, I will show some code examples, but you do not need to write the code yourself.

Each type can have a single **finalizer** that will be called by the .NET runtime when the resources need to be released. A finalizer has the same name as a constructor; that is, the name of the type, but it is prefixed with a tilde, `~`, as shown in the following code:

```
public class ObjectWithUnmanagedResources
{
  public ObjectWithUnmanagedResources() // constructor
  {
    // allocate any unmanaged resources
  }
  ~ObjectWithUnmanagedResources() // Finalizer aka destructor
  {
    // deallocate any unmanaged resources
  }
}
```

Do not confuse a finalizer (also known as a **destructor**) with a `Deconstruct` method. A destructor releases resources; that is, it destroys an object in memory. A `Deconstruct` method returns an object split up into its constituent parts and uses the C# deconstruction syntax, for example, when working with tuples. See *Chapter 5*, *Building Your Own Types with Object-Oriented Programming*, for details of `Deconstruct` methods.

The preceding code example is the minimum you should do when working with unmanaged resources. But the problem with only providing a finalizer is that the .NET garbage collector requires two garbage collections to completely release the allocated resources for this type.

Though optional, it is recommended to also provide a method to allow a developer who uses your type to explicitly release resources. This would allow the garbage collector to release managed parts of an unmanaged resource,

such as a file, immediately and deterministically. This would mean it releases the managed memory part of the object in a single garbage collection instead of two rounds of garbage collection.

There is a standard mechanism for doing this by implementing the `IDisposable` interface, as shown in the following example:

```
public class ObjectWithUnmanagedResources : IDisposable
{
  public ObjectWithUnmanagedResources()
  {
    // allocate unmanaged resource
  }
  ~ObjectWithUnmanagedResources() // Finalizer
  {
    Dispose(false);
  }
  bool disposed = false; // have resources been released?
  public void Dispose()
  {
    Dispose(true);
    // tell garbage collector it does not need to call the finalizer
    GC.SuppressFinalize(this);
  }
  protected virtual void Dispose(bool disposing)
  {
    if (disposed) return;
    // deallocate the *unmanaged* resource
    // ...
    if (disposing)
    {
      // deallocate any other *managed* resources
      // ...
    }
    disposed = true;
  }
}
```

There are two `Dispose` methods, one `public` and one `protected`:

- The `public void Dispose` method will be called by a developer using your type. When called, both unmanaged and managed resources need to be deallocated.
- The `protected virtual void Dispose` method with a `bool` parameter is used internally to implement the deallocation of resources. It needs to check the `disposing` parameter and `disposed` field because if the finalizer thread has already run and it called the `~ObjectWithUnmanagedResources` method, then only unmanaged resources need to be deallocated.

The call to `GC.SuppressFinalize(this)` is what notifies the garbage collector that it no longer needs to run the finalizer, and removes the need for a second garbage collection.

Ensuring that Dispose is called

When someone uses a type that implements `IDisposable`, they can ensure that the public `Dispose` method is called with the `using` statement, as shown in the following code:

```
using (ObjectWithUnmanagedResources thing = new())
{
  // code that uses thing
}
```

The compiler converts your code into something like the following, which guarantees that even if an exception occurs, the `Dispose` method will still be called:

```
ObjectWithUnmanagedResources thing = new();
try
{
```

```
  // code that uses thing
}
finally
{
  if (thing != null) thing.Dispose();
}
```

You will see practical examples of releasing unmanaged resources with `IDisposable`, `using` statements, and `try` ... `finally` blocks in *Chapter 9*, *Working with Files, Streams, and Serialization*.

Working with null values

You have seen how reference types and different from value types in how they are stored in memory, and how to store primitive values like numbers in `struct` variables. But what if a variable does not yet have a value? How can we indicate that? C# has the concept of a `null` value, which can be used to indicate that a variable has not been set.

Making a value type nullable

By default, **value types** like `int` and `DateTime` must always have a *value*, hence their name. Sometimes, for example, when reading values stored in a database that allows empty, missing, or `null` values, it is convenient to allow a value type to be `null`. We call this a **nullable value type**.

You can enable this by adding a question mark as a suffix to the type when declaring a variable.

Let's see an example:

1. Use your preferred coding tool to add a new **Console App** / `console` project named `NullHandling` to the `Chapter06` workspace/solution.
   - In Visual Studio Code, select `NullHandling` as the active OmniSharp project.
   - In Visual Studio 2022, set `NullHandling` as the startup project.
2. In `NullHandling.csproj`, add an `<ItemGroup>` to globally and statically import the `System.Console` class.
3. In `Program.cs`, delete the existing statements and then add statements to declare and assign values, including `null`, to `int` variables, one suffixed with `?` and one not, as shown in the following code:

```
int thisCannotBeNull  = 4;
thisCannotBeNull = null; // compile error!
WriteLine(thisCannotBeNull);
int? thisCouldBeNull = null;
WriteLine(thisCouldBeNull);
WriteLine(thisCouldBeNull.GetValueOrDefault());
thisCouldBeNull = 7;
WriteLine(thisCouldBeNull);
WriteLine(thisCouldBeNull.GetValueOrDefault());
```

1. Build the project and note the compile error, as shown in the following output:

```
Cannot convert null to 'int' because it is a non-nullable value type
```

1. Comment out the statement that gives the compile error.
2. Run the code and view the result, as shown in the following output:

```
0
7
7
```

The first line is blank because it is outputting the `null` value!

1. Add statements to use an alternative syntax, as shown in the following code:

```
// the actual type of int? is Nullable<int>
Nullable<int> thisCouldAlsoBeNull = null;
thisCouldAlsoBeNull = 9;
WriteLine(thisCouldAlsoBeNull);
```

1. Click in `Nullable<int>` and press F12, or right-click and choose **Go To Definition.**
2. Note the generic value type, `Nullable<T>`, must have a type `T` that is a `struct` aka value type, and it has useful members like `HasValue`, `Value`, and `GetValueOrDefault`, as shown in *Figure 6.2*:

Figure 6.2: Revealing Nullable<T> members

Good Practice: When you append a `?` after a `struct` aka value type, you are changing it to a different `struct` aka value type.

Understanding null-related initialisms

Before we see some code, let's review some commonly used initialisms, as shown in the following table:

Initialism	Meaning	Description
NRT	Nullable Reference Types	A compiler feature introduced with C# 8 and enabled by default in new projects with C# 10 that performs static analysis of your code at design time and shows warnings of potential misuse of `null` values for reference types.
NRE	`NullReferenceException`	An exception thrown at runtime when **dereferencing** a `null` value, aka accessing a variable or member that has a `null` value.
ANE	`ArgumentNullException`	An exception thrown at runtime by a method invocation when an argument has a `null` value when that is not valid.

Understanding nullable reference types

The use of the `null` value is so common, in so many languages, that many experienced programmers never question the need for its existence. But there are many scenarios where we could write better, simpler code if a variable is not allowed to have a `null` value.

The most significant change to the C# 8 language compiler was the introduction of checks and warnings for nullable and non-nullable reference types. "But wait!", you are probably thinking, "Reference types are already nullable!"

And you would be right, but in C# 8 and later, reference types can be configured to no longer allow the `null` value by setting a file- or project-level option to enable this useful new feature. Since this is a big change for C#, Microsoft decided to make the feature opt-in.

It will take multiple years for this new C# language compiler feature to make an impact since thousands of existing library packages and apps will expect the old behavior. Even Microsoft did not have time to fully implement this new feature in all the main .NET packages until .NET 6. Important libraries like `Microsoft.Extensions` for logging, dependency injections, and configuration were not annotated until .NET 7.

During the transition, you can choose between several approaches for your own projects:

- **Default**: For projects created using .NET 5 or earlier, no changes are needed. Non-nullable reference types are not checked. For projects created using .NET 6 or later, nullability checks are enabled by default, but this can be disabled by either deleting the `<Nullable>` entry in the project file or setting it to `disable`.
- **Opt-in project, opt-out files**: Enable the feature at the project level and, for any files that need to remain compatible with old behavior, opt out. This was the approach Microsoft was using internally while it updated its own packages to use this new feature.
- **Opt-in files**: Only enable the feature for individual files.

Controlling the nullability warning check feature

To enable the nullability warning check feature at the project level, add the following to your project file:

```
<PropertyGroup>
  ...
  <Nullable>enable</Nullable>
</PropertyGroup>
```

To disable the nullability warning check feature at the project level, add the following to your project file:

```
<PropertyGroup>
  ...
  <Nullable>disable</Nullable>
</PropertyGroup>
```

To disable the feature at the file level, add the following to the top of a code file:

```
#nullable disable
```

To enable the feature at the file level, add the following to the top of a code file:

```
#nullable enable
```

Declaring non-nullable variables and parameters

If you enable nullable reference types and you want a reference type to be assigned the `null` value, then you will have to use the same syntax as making a value type nullable, that is, adding a `?` symbol after the type declaration.

So, how do nullable reference types work? Let's look at an example. When storing information about an address, you might want to force a value for the street, city, and region, but the building can be left blank, that is, `null`:

1. In `NullHandling.csproj`, add a class file named `Address.cs`.
2. in `Address.cs`, add statements to declare an `Address` class with four fields, as shown in the following code:

```
public class Address
{
  public string? Building;
  public string Street;
  public string City;
  public string Region;
}
```

1. After a few seconds, note the warnings about non-nullable fields, like `Street` not being initialized, as shown in *Figure 6.3*:

Figure 6.3: Warning messages about non-nullable fields in the Error List window

2. Assign the empty `string` value to each of the three fields that are non-nullable, as shown in the following code:

```
public string Street = string.Empty;
public string City = string.Empty;
public string Region = string.Empty;
```

1. In `Program.cs`, add statements to instantiate an `Address` and set its properties, as shown in the following code:

```
Address address = new()
{
  Building = null,
  Street = null,
  City = "London",
  Region = "UK"
};
```

1. Note the `Warning CS8625` on setting the `Street` but not the `Building`, as shown in the following output:

```
Cannot convert null literal to non-nullable reference type.
```

1. Append an exclamation mark after `null` when setting `Street`, as shown highlighted in the following code:

```
Street = null!, // null-forgiving operator
```

1. Note the warning disappears.
2. Add statements that would dereference the `Building` and `Street` properties, as shown in the following code:

```
WriteLine(address.Building.Length);
WriteLine(address.Street.Length);
```

1. Note the `Warning CS8602` on setting the `Building` but not the `Street`, as shown in the following output:

```
Dereference of a possibly null reference.
```

At runtime it is still possible for an exception to be thrown when working with `Street`, but the compiler should continue to warn you of potential exceptions when working with `Building` so you can change your code to avoid them.

1. Use the null-conditional operator to return `null` instead of accessing the `Length`, as shown in the following code:

```
WriteLine(address.Building?.Length);
```

1. Run the console app and note that the statement that accesses the `Length` of the `Building` outputs a `null` value (blank line), but a runtime exception occurs when we access the `Length` of the `Street`, as shown in the following output:

```
Unhandled exception. System.NullReferenceException: Object reference not set to an instance of a
```

It is worth reminding yourself that NRT is only about asking the compiler to provide warnings about potential `null` values that might cause problems. It does not actually change the behavior of your code. It performs a static analysis of your code at compile time.

So, this explains why the new language feature is named nullable reference types. Starting with C# 8.0, unadorned reference types can become non-nullable, and the same syntax is used to make a reference type nullable as is used for value types.

> Suffixing a reference type with `?` does not change the type. This is different from suffixing a value type with `?` that changes its type to `Nullable<T>`. Reference types can already have `null` values. All you are doing with nullable reference types (NRTs) is telling the compiler that you expect it could be `null`, so the compiler does not need to warn you. But this does not remove the need to perform `null` checks throughout your code.

Now let's see language features for working with `null` values that do change the behavior of your code and work well as a complement to NRTs.

Checking for null

Checking whether a nullable reference type or nullable value type variable currently contains `null` is important because if you do not, a `NullReferenceException` can be thrown, which results in an error. You should check for a `null` value before using a nullable variable, as shown in the following code:

```
// check that the variable is not null before using it
if (thisCouldBeNull != null)
{
  // access a member of thisCouldBeNull
  int length = thisCouldBeNull.Length; // could throw exception
```

```
  ...
}
```

C# 7 introduced `is` combined with the `!` (not) operator as an alternative to `!=`, as shown in the following code:

```
if (!(thisCouldBeNull is null))
{
```

C# 9 introduced `is not` as an even clearer alternative, as shown in the following code:

```
if (thisCouldBeNull is not null)
{
```

If you are trying to use a member of a variable that might be `null`, use the **null-conditional operator**, `?.`, as shown in the following code:

```
string authorName = null;
// the following throws a NullReferenceException
int x = authorName.Length;
// instead of throwing an exception, null is assigned to y
int? y = authorName?.Length;
```

Sometimes you want to either assign a variable to a result or use an alternative value, such as `3`, if the variable is `null`. You do this using the **null-coalescing operator**, `??`, as shown in the following code:

```
// result will be 3 if authorName?.Length is null
int result = authorName?.Length ?? 3;
Console.WriteLine(result);
```

> **Good Practice**: Even if you enable nullable reference types, you should still check non-nullable parameters for `null` and throw an `ArgumentNullException`.

Checking for null in method parameters

When defining methods with parameters, it is good practice to check for `null` values.

In earlier versions of C#, you would have to write `if` statements to check for `null` parameter values and then throw an `ArgumentNullException` for any parameter that is `null`, as shown in the following code:

```
public void Hire(Person manager, Person employee)
{
  if (manager == null)
  {
    throw new ArgumentNullException(nameof(manager));
  }
  if (employee == null)
  {
    throw new ArgumentNullException(nameof(employee));
  }
  ...
}
```

C# 10 introduced a convenience method to throw an exception if an argument is `null`, as shown in the following code:

```
public void Hire(Person manager, Person employee)
{
  ArgumentNullException.ThrowIfNull(manager);
  ArgumentNullException.ThrowIfNull(employee);
  ...
}
```

C# 11 introduced a new `!!` operator that does this for you, as shown in the following code:

```
public void Hire(Person manager!!, Person employee!!)
{
  ...
}
```

The `if` statement and throwing of the exception are done for you. The code is injected and executes before any statements that you write.

This syntax is controversial within the C# developer community. Some would prefer the use of attributes to decorate parameters instead of a pair of characters. The .NET product team claims to have saved more than 10,000 lines of code throughout the .NET libraries by using this feature. That sounds like a good reason to use it to me! And no one must use it if they choose not to.

Inheriting from classes

The `Person` type we created earlier derived (inherited) from `object`, the alias for `System.Object`. Now, we will create a subclass that inherits from `Person`:

1. In the `PacktLibrary` project, add a new class file named `Employee.cs`.
2. Modify its contents to define a class named `Employee` that derives from `Person`, as shown in the following code:

```
namespace Packt.Shared;
public class Employee : Person
{
}
```

1. In the `PeopleApp` project, in `Program.cs`, add statements to create an instance of the `Employee` class, as shown in the following code:

```
Employee john = new()
{
  Name = "John Jones",
  DateOfBirth = new(year: 1990, month: 7, day: 28)
};
john.WriteToConsole();
```

1. Run the code and view the result, as shown in the following output:

```
John Jones was born on a Saturday.
```

Note that the `Employee` class has inherited all the members of `Person`.

Extending classes to add functionality

Now, we will add some employee-specific members to extend the class.

1. In `Employee.cs`, add statements to define two properties, for an employee code and the date they were hired, as shown in the following code:

```
public string? EmployeeCode { get; set; }
public DateTime HireDate { get; set; }
```

1. In `Program.cs`, add statements to set John's employee code and hire date, as shown in the following code:

```
john.EmployeeCode = "JJ001";
john.HireDate = new(year: 2014, month: 11, day: 23);
WriteLine($"{john.Name} was hired on {john.HireDate:dd/MM/yy}");
```

1. Run the code and view the result, as shown in the following output:

```
John Jones was hired on 23/11/14
```

Hiding members

So far, the `WriteToConsole` method is inherited from `Person`, and it only outputs the employee's name and date of birth. We might want to change what this method does for an employee:

1. In `Employee.cs`, add statements to redefine the `WriteToConsole` method, as shown highlighted in the following code:

```
namespace Packt.Shared;
public class Employee : Person
{
  public string? EmployeeCode { get; set; }
  public DateTime HireDate { get; set; }
  public void WriteToConsole()
  {
    WriteLine(format:
      "{0} was born on {1:dd/MM/yy} and hired on {2:dd/MM/yy}",
      arg0: Name,
      arg1: DateOfBirth,
      arg2: HireDate);
  }
}
```

1. Run the code and view the result, as shown in the following output:

```
John Jones was born on 28/07/90 and hired on 01/01/01
John Jones was hired on 23/11/14
```

Your coding tool warns you that your method now hides the method from `Person` by drawing a squiggle under the method name; the **PROBLEMS/Error List** window includes more details; and the compiler will output the warning when you build and run the console application, as shown in *Figure 6.4*:

Figure 6.4: Hidden method warning

As the warning describes, you can hide this message by applying the `new` keyword to the method, to indicate that you are deliberately replacing the old method, as shown highlighted in the following code:

```
public new void WriteToConsole()
```

Overriding members

Rather than hiding a method, it is usually better to **override** it. You can only override if the base class chooses to allow overriding, by applying the `virtual` keyword to any methods that should allow overriding.

Let's see an example:

1. In `Program.cs`, add a statement to write the value of the `john` variable to the console using its `string` representation, as shown in the following code:

```
WriteLine(john.ToString());
```

1. Run the code and note that the `ToString` method is inherited from `System.Object`, so the implementation returns the namespace and type name, as shown in the following output:

```
Packt.Shared.Employee
```

1. In `Person.cs` (not in the `Employee` class!), override this behavior by adding a `ToString` method to output the name of the person as well as the type name, as shown in the following code:

```
// overridden methods
public override string ToString()
{
  return $"{Name} is a {base.ToString()}";
}
```

The `base` keyword allows a subclass to access members of its superclass; that is, the **base class** that it inherits or derives from.

1. Run the code and view the result. Now, when the `ToString` method is called, it outputs the person's name, as well as returning the base class's implementation of `ToString`, as shown in the following output:

```
John Jones is a Packt.Shared.Employee
```

Good Practice: Many real-world APIs, for example, Microsoft's Entity Framework Core, Castle's DynamicProxy, and Optimizely CMS's content models, require the properties that you define in your classes to be marked as `virtual` so that they can be overridden. Carefully decide which of your method and property members should be marked as `virtual`.

Inheriting from abstract classes

Earlier in this chapter, you learned about interfaces that can define a set of members that a type must have to meet a basic level of functionality. These are very useful, but their main limitation is that until C# 8 they could not provide any implementation of their own.

This is a particular problem if you still need to create class libraries that will work with .NET Framework and other platforms that do not support .NET Standard 2.1.

In those earlier platforms, you could use abstract classes as a sort of halfway house between a pure interface and a fully implemented class.

When a class is marked as `abstract`, this means that it cannot be instantiated because you are indicating that the class is not complete. It needs more implementation before it can be instantiated. For example, the `System.IO.Stream` class is `abstract` because it implements common functionality that all streams would need but is not complete, so you cannot instantiate it using `new Stream()`.

Let's compare the two types of interface and two types of class, as shown in the following code:

```
public interface INoImplementation // C# 1.0 and later
{
  void Alpha(); // must be implemented by derived type
}
public interface ISomeImplementation // C# 8.0 and later
{
  void Alpha(); // must be implemented by derived type
  void Beta()
  {
    // default implementation; can be overridden
  }
}
public abstract class PartiallyImplemented // C# 1.0 and later
{
  public abstract void Gamma(); // must be implemented by derived type
  public virtual void Delta() // can be overridden
  {
    // implementation
  }
}
public class FullyImplemented : PartiallyImplemented, ISomeImplementation
{
```

```
  public void Alpha()
  {
    // implementation
  }
  public override void Gamma()
  {
    // implementation
  }
}
// you can only instantiate the fully implemented class
FullyImplemented a = new();
// all the other types give compile errors
PartiallyImplemented b = new(); // compile error!
ISomeImplementation c = new(); // compile error!
INoImplementation d = new(); // compile error!
```

Preventing inheritance and overriding

You can prevent another developer from inheriting from your class by applying the `sealed` keyword to its definition. No one can inherit from Scrooge McDuck, as shown in the following code:

```
public sealed class ScroogeMcDuck
{
}
```

An example of `sealed` in .NET is the `string` class. Microsoft has implemented some extreme optimizations inside the `string` class that could be negatively affected by your inheritance, so Microsoft prevents that.

You can prevent someone from further overriding a `virtual` method in your class by applying the `sealed` keyword to the method. No one can change the way Lady Gaga sings, as shown in the following code:

```
namespace Packt.Shared;
public class Singer
{
  // virtual allows this method to be overridden
  public virtual void Sing()
  {
    WriteLine("Singing...");
  }
}
public class LadyGaga : Singer
{
  // sealed prevents overriding the method in subclasses
  public sealed override void Sing()
  {
    WriteLine("Singing with style...");
  }
}
```

You can only seal an overridden method.

Understanding polymorphism

You have now seen two ways to change the behavior of an inherited method. We can *hide* it using the `new` keyword (known as **non-polymorphic inheritance**), or we can *override* it (known as **polymorphic inheritance**).

Both ways can access members of the base or superclass by using the `base` keyword, so what is the difference?

It all depends on the type of variable holding a reference to the object. For example, a variable of the `Person` type can hold a reference to a `Person` class, or any type that derives from `Person`.

Let's see how this could affect your code:

1. In `Employee.cs`, add statements to override the `ToString` method so it writes the employee's name and code to the console, as shown in the following code:

```
public override string ToString()
{
  return $"{Name}'s code is {EmployeeCode}";
}
```

1. In `Program.cs`, write statements to create a new employee named Alice, store it in a variable of type `Person`, and call both variables' `WriteToConsole` and `ToString` methods, as shown in the following code:

```
Employee aliceInEmployee = new()
  { Name = "Alice", EmployeeCode = "AA123" };
Person aliceInPerson = aliceInEmployee;
aliceInEmployee.WriteToConsole();
aliceInPerson.WriteToConsole();
WriteLine(aliceInEmployee.ToString());
WriteLine(aliceInPerson.ToString());
```

1. Run the code and view the result, as shown in the following output:

```
Alice was born on 01/01/01 and hired on 01/01/01
Alice was born on a Monday
Alice's code is AA123
Alice's code is AA123
```

When a method is hidden with `new`, the compiler is not smart enough to know that the object is an `Employee`, so it calls the `WriteToConsole` method in `Person`.

When a method is overridden with `virtual` and `override`, the compiler is smart enough to know that although the variable is declared as a `Person` class, the object itself is an `Employee` class and, therefore, the `Employee` implementation of `ToString` is called.

The member modifiers and the effect they have are summarized in the following table:

Variable type	Member modifier	Method executed	In class
Person		WriteToConsole	Person
Employee	new	WriteToConsole	Employee
Person	virtual	ToString	Employee
Employee	override	ToString	Employee

In my opinion, polymorphism is academic to most programmers. If you get the concept, that's cool; but, if not, I suggest that you don't worry about it. Some people like to make others feel inferior by saying understanding polymorphism is important for all C# programmers, but IMHO it's not.

You can have a successful career with C# and never need to be able to explain polymorphism, just as a racing car driver doesn't need to be able to explain the engineering behind fuel injection.

> **Good Practice**: You should use `virtual` and `override` rather than `new` to change the implementation of an inherited method whenever possible.

Casting within inheritance hierarchies

Casting between types is subtly different from converting between types. Casting is between similar types, like between a 16-bit integer and a 32-bit integer, or between a superclass and one of its subclasses. **Converting** is between dissimilar types, such as between text and a number.

For example, if you need to work with multiple types of stream, then instead of declaring specific types of stream like `MemoryStream` or `FileStream`, you could declare an array of `Stream`, the supertype of `MemoryStream` and `FileStream`.

Implicit casting

In the previous example, you saw how an instance of a derived type can be stored in a variable of its base type (or its base's base type, and so on). When we do this, it is called **implicit casting**.

Explicit casting

Going the other way is an explicit cast, and you must use parentheses around the type you want to cast into as a prefix to do it:

1. In `Program.cs`, add a statement to assign the `aliceInPerson` variable to a new `Employee` variable, as shown in the following code:

```
Employee explicitAlice = aliceInPerson;
```

1. Your coding tool displays a red squiggle and a compile error, as shown in *Figure 6.5*:

Figure 6.5: A missing explicit cast compile error

2. Change the statement to prefix the assigned variable named with a cast to the `Employee` type, as shown in the following code:

```
Employee explicitAlice = (Employee)aliceInPerson;
```

Avoiding casting exceptions

The compiler is now happy; but, because `aliceInPerson` might be a different derived type, like `Student` instead of `Employee`, we need to be careful. In a real application with more complex code, the current value of this variable could have been set to a `Student` instance, and then this statement would throw an `InvalidCastException` error.

Using is to check a type

We can handle this by writing a `try` statement, but there is a better way. We can check the type of an object using the `is` keyword:

1. Wrap the explicit cast statement in an `if` statement, as shown highlighted in the following code:

```
if (aliceInPerson is Employee)
{
  WriteLine($"{nameof(aliceInPerson)} IS an Employee");
  Employee explicitAlice = (Employee)aliceInPerson;
  // safely do something with explicitAlice
}
```

1. Run the code and view the result, as shown in the following output:

```
aliceInPerson IS an Employee
```

You could simplify the code further using a declaration pattern and this will avoid needing to perform an explicit cast, as shown in the following code:

```
if (aliceInPerson is Employee explicitAlice)
{
  WriteLine($"{nameof(aliceInPerson)} IS an Employee");
  // safely do something with explicitAlice
}
```

What if you want to execute a block of statements when Alice is *not* an employee?

In the past, you would have had to use the `!` (not) operator, as shown in the following code:

```
if (!(aliceInPerson is Employee))
```

With C# 9 and later, you can use the `not` keyword, as shown in the following code:

```
if (aliceInPerson is not Employee)
```

Using as to cast a type

Alternatively, you can use the `as` keyword to cast. Instead of throwing an exception, the `as` keyword returns `null` if the type cannot be cast.

1. In `Program.cs`, add statements to cast Alice using the `as` keyword and then check whether the return value is not null, as shown in the following code:

```
Employee? aliceAsEmployee = aliceInPerson as Employee; // could be null
if (aliceAsEmployee is not null)
```

```
{
  WriteLine($"{nameof(aliceInPerson)} AS an Employee");
  // safely do something with aliceAsEmployee
}
```

Since accessing a member of a `null` variable will throw a `NullReferenceException` error, you should always check for `null` before using the result.

1. Run the code and view the result, as shown in the following output:

```
aliceInPerson AS an Employee
```

Good Practice: Use the `is` and `as` keywords to avoid throwing exceptions when casting between derived types. If you don't do this, you must write `try-catch` statements for `InvalidCastException`.

Inheriting and extending .NET types

.NET has prebuilt class libraries containing hundreds of thousands of types. Rather than creating your own completely new types, you can often get a head start by deriving from one of Microsoft's types to inherit some or all its behavior and then overriding or extending it.

Inheriting exceptions

As an example of inheritance, we will derive a new type of exception:

1. In the `PacktLibrary` project, add a new class file named `PersonException.cs`.
2. Modify the contents of the file to define a class named `PersonException` with three constructors, as shown in the following code:

```
namespace Packt.Shared;
public class PersonException : Exception
{
  public PersonException() : base() { }
  public PersonException(string message) : base(message) { }
  public PersonException(string message, Exception innerException)
    : base(message, innerException) { }
}
```

Unlike ordinary methods, constructors are not inherited, so we must explicitly declare and explicitly call the `base` constructor implementations in `System.Exception` (or whichever exception class you derived from) to make them available to programmers who might want to use those constructors with our custom exception.

1. In `Person.cs`, add statements to define a method that throws an exception if a date/time parameter is earlier than a person's date of birth, as shown in the following code:

```
public void TimeTravel(DateTime when)
{
  if (when <= DateOfBirth)
  {
    throw new PersonException("If you travel back in time to a date earlier than your own birth,
  }
  else
  {
    WriteLine($"Welcome to {when:yyyy}!");
  }
}
```

1. In `Program.cs`, add statements to test what happens when employee John Jones tries to time travel too far back, as shown in the following code:

```
try
{
  john.TimeTravel(when: new(1999, 12, 31));
```

```
  john.TimeTravel(when: new(1950, 12, 25));
}
catch (PersonException ex)
{
  WriteLine(ex.Message);
}
```

1. Run the code and view the result, as shown in the following output:

```
Welcome to 1999!
If you travel back in time to a date earlier than your own birth, then the universe will explode
```

Good Practice: When defining your own exceptions, give them the same three constructors that explicitly call the built-in ones in `System.Exception`. Other exceptions that you might inherit from may have more.

Extending types when you can't inherit

Earlier, we saw how the `sealed` modifier can be used to prevent inheritance.

Microsoft has applied the `sealed` keyword to the `System.String` class so that no one can inherit and potentially break the behavior of strings.

Can we still add new methods to strings? Yes, if we use a language feature named **extension methods**, which was introduced with C# 3.0. To properly understand extension methods, we need to review static methods first.

Using static methods to reuse functionality

Since the first version of C#, we've been able to create `static` methods to reuse functionality, such as the ability to validate that a `string` contains an email address. The implementation will use a regular expression that you will learn more about in *Chapter 8*, *Working with Common .NET Types*.

Let's write some code:

1. In the `PacktLibrary` project, add a new class file named `StringExtensions.cs`.
2. Modify `StringExtensions.cs`, as shown in the following code, and note the following:
   - The class imports a namespace for handling regular expressions.
   - The `IsValidEmail` method is `static` and it uses the `Regex` type to check for matches against a simple email pattern that looks for valid characters before and after the `@` symbol.

```
using System.Text.RegularExpressions; // to get Regex
namespace Packt.Shared;
public class StringExtensions
{
  public static bool IsValidEmail(string input)
  {
    // use a simple regular expression to check
    // that the input string is a valid email
    return Regex.IsMatch(input,
      @"[a-zA-Z0-9\.-_]+@[a-zA-Z0-9\.-_]+");
  }
}
```

1. In `Program.cs`, add statements to validate two examples of email addresses, as shown in the following code:

```
string email1 = "pamela@test.com";
string email2 = "ian&test.com";
WriteLine("{0} is a valid e-mail address: {1}",
  arg0: email1,
  arg1: StringExtensions.IsValidEmail(email1));
WriteLine("{0} is a valid e-mail address: {1}",
  arg0: email2,
  arg1: StringExtensions.IsValidEmail(email2));
```

1. Run the code and view the result, as shown in the following output:

```
pamela@test.com is a valid e-mail address: True
ian&test.com is a valid e-mail address: False
```

This works, but extension methods can reduce the amount of code we must type and simplify the usage of this function.

Using extension methods to reuse functionality

It is easy to make `static` methods into extension methods:

1. In `StringExtensions.cs`, add the `static` modifier before the class, and add the `this` modifier before the `string` type, as highlighted in the following code:

```
public static class StringExtensions
{
  public static bool IsValidEmail(this string input)
  {
```

These two changes tell the compiler that it should treat the method as one that extends the string type.

1. In `Program.cs`, add statements to use the extension method for `string` values that need to be checked for valid email addresses, as shown in the following code:

```
WriteLine("{0} is a valid e-mail address: {1}",
  arg0: email1,
  arg1: email1.IsValidEmail());
WriteLine("{0} is a valid e-mail address: {1}",
  arg0: email2,
  arg1: email2.IsValidEmail());
```

Note the subtle simplification in the syntax for calling the `IsValidEmail` method. The older, longer syntax still works too.

1. The `IsValidEmail` extension method now appears to be a method just like all the actual instance methods of the `string` type, such as `IsNormalized`, except with a small down arrow on the method icon to indicate an extension method, as shown in *Figure 6.6*:

Figure 6.6: Extension methods appear in IntelliSense alongside instance methods

2. Run the code and view the result, which will be the same as before.

Good Practice: Extension methods cannot replace or override existing instance methods. You cannot, for example, redefine the `Insert` method. The extension method will appear as an overload in IntelliSense, but an instance method will be called in preference to an extension method with the same name and signature.

Although extension methods might not seem to give a big benefit, in *Chapter 11*, *Querying and Manipulating Data Using LINQ*, you will see some extremely powerful uses of extension methods.

Using an analyzer to write better code

We have now spent many chapters learning how to write C# code. Before we move on to learning about the .NET libraries, let's see how we can get help to write better code.

.NET analyzers find potential issues and suggest fixes for them. **StyleCop** is a commonly used analyzer for helping you write better C# code.

Let's see it in action:

1. Use your preferred code editor to add a **Console App** / `console` project named `CodeAnalyzing` to the `Chapter06` solution/workspace.
2. In the `CodeAnalyzing` project, add a package reference for `StyleCop.Analyzers`.
3. Add a JSON file to your project named `stylecop.json` for controlling StyleCop settings.
4. Modify its contents, as shown in the following markup:

```
{
  "$schema": "https://raw.githubusercontent.com/DotNetAnalyzers/StyleCopAnalyzers/master/StyleCop
  "settings": {
  }
}
```

The `$schema` entry enables IntelliSense while editing the `stylecop.json` file in your code editor.

1. Move the insertion point inside the `settings` section and press Ctrl + Space, and note the IntelliSense showing valid subsections of settings, as shown in *Figure 6.7*:

Figure 6.7: stylecop.json IntelliSense showing valid subsections of settings

2. In the `CodeAnalyzing` project file, add entries to configure the file named `stylecop.json` to not be included in published deployments, and to enable it as an additional file for processing during development, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net7.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="StyleCop.Analyzers" Version="1.2.0-*">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers</IncludeAssets>
```

```
    </PackageReference>
  </ItemGroup>
  <ItemGroup>
    <None Remove="stylecop.json" />
  </ItemGroup>
  <ItemGroup>
    <AdditionalFiles Include="stylecop.json" />
  </ItemGroup>
</Project>
```

1. In `Program.cs`, delete the existing statements and then add some statements to explicitly define the `Program` class with its `Main` method, as shown in the following code:

```
using System.Diagnostics;
namespace CodeAnalyzing;
class Program
{
  static void Main(string[] args)
  {
    Debug.WriteLine("Hello, Debugger!");
  }
}
```

1. Build the `CodeAnalyzing` project.

2. You will see warnings for everything it thinks is wrong, as shown in *Figure 6.8*:

Figure 6.8: StyleCop code analyzer warnings

3. For example, it wants `using` directives to be put within the namespace declaration, as shown in the following output:

```
C:\cs11dotnet7\Chapter06\CodeAnalyzing\Program.cs(1,1): warning SA1200: Using directive should ap
```

Suppressing warnings

To suppress a warning, you have several options, including adding code and setting configuration.

To suppress a warning using an attribute, add an assembly-level attribute, as shown in the following code:

```
[assembly:SuppressMessage("StyleCop.CSharp.OrderingRules", "SA1200:UsingDirectivesMustBePlacedWi
```

To suppress a warning using a directive, add `#pragma` statements around the statement that is causing the warning, as shown in the following code:

```
#pragma warning disable SA1200 // UsingDirectivesMustBePlacedWithinNamespace
using System.Diagnostics;
#pragma warning restore SA1200 // UsingDirectivesMustBePlacedWithinNamespace
```

Let's suppress the warning by modifying the `stylecop.json` file:

1. In `stylecop.json`, add a configuration option to set `using` statements to be allowable outside a namespace, as shown highlighted in the following markup:

```
{
  "$schema": "https://raw.githubusercontent.com/DotNetAnalyzers/StyleCopAnalyzers/master/StyleCop
  "settings": {
    "orderingRules": {
      "usingDirectivesPlacement": "outsideNamespace"
    }
  }
}
```

1. Build the project and note that warning SA1200 has disappeared.
2. In `stylecop.json`, set the using directives placement to `preserve`, which allows `using` statements both inside and outside a namespace, as shown highlighted in the following markup:

```
"orderingRules": {
  "usingDirectivesPlacement": "preserve"
}
```

Fixing the code

Now, let's fix all the other warnings:

1. In `CodeAnalyzing.csproj`, add an element to automatically generate an XML file for documentation, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net7.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <GenerateDocumentationFile>true</GenerateDocumentationFile>
  </PropertyGroup>
```

1. In `stylecop.json`, add a configuration option to provide values for documentation for the company name and copyright text, as shown highlighted in the following markup:

```
{
  "$schema": "https://raw.githubusercontent.com/DotNetAnalyzers/StyleCopAnalyzers/master/StyleCop
  "settings": {
    "orderingRules": {
      "usingDirectivesPlacement": "preserve"
    },
    "documentationRules": {
      "companyName": "Packt",
      "copyrightText": "Copyright (c) Packt. All rights reserved."
    }
  }
}
```

1. In `Program.cs`, add comments for a file header with company and copyright text, move the `using System;` declaration inside the namespace, and set explicit access modifiers and XML comments for the class and method, as shown in the following code:

```
// <copyright file="Program.cs" company="Packt">
// Copyright (c) Packt. All rights reserved.
// </copyright>
```

```
namespace CodeAnalyzing;
using System.Diagnostics;
/// <summary>
/// The main class for this console app.
/// </summary>
public class Program
{
  /// <summary>
  /// The main entry point for this console app.
  /// </summary>
  /// <param name="args">
  /// A string array of arguments passed to the console app.
  /// </param>
  public static void Main(string[] args)
  {
    Debug.WriteLine("Hello, Debugger!");
  }
}
```

1. Build the project.
2. Expand the `bin/Debug/net7.0` folder (remember to **Show All Files** if you are using Visual Studio 2022) and note the autogenerated file named `CodeAnalyzing.xml`, as shown in the following markup:

```
<?xml version="1.0"?>
<doc>
    <assembly>
        <name>CodeAnalyzing</name>
    </assembly>
    <members>
        <member name="T:CodeAnalyzing.Program">
            <summary>
            The main class for this console app.
            </summary>
        </member>
        <member name="M:CodeAnalyzing.Program.Main(System.String[])">
            <summary>
            The main entry point for this console app.
            </summary>
            <param name="args">
            A string array of arguments passed to the console app.
            </param>
        </member>
    </members>
</doc>
```

The `CodeAnalyzing.xml` file can then be processed by a tool like DocFX to convert it into documentation files, as shown at the following link: https://www.jamescroft.co.uk/building-net-project-docs-with-docfx-on-github-pages/

Understanding common StyleCop recommendations

Inside a code file, you should order the contents as shown in the following list:

1. External alias directives
2. Using directives
3. Namespaces
4. Delegates
5. Enums
6. Interfaces
7. Structs
8. Classes

Within a class, record, struct, or interface, you should order the contents as shown in the following list:

1. Fields
2. Constructors

3. Destructors (finalizers)
4. Delegates
5. Events
6. Enums
7. Interfaces
8. Properties
9. Indexers
10. Methods
11. Structs
12. Nested classes and records

Good Practice: You can learn about all the StyleCop rules at the following link: https://github.com/DotNetAnalyzers/StyleCopAnalyzers/blob/master/DOCUMENTATION.md.

Practicing and exploring

Test your knowledge and understanding by answering some questions, getting some hands-on practice, and exploring this chapter's topics with more in-depth research.

Exercise 6.1 – Test your knowledge

Answer the following questions:

1. What is a delegate?
2. What is an event?
3. How are a base class and a derived class related, and how can the derived class access the base class?
4. What is the difference between `is` and `as` operators?
5. Which keyword is used to prevent a class from being derived from or a method from being further overridden?
6. Which keyword is used to prevent a class from being instantiated with the `new` keyword?
7. Which keyword is used to allow a member to be overridden?
8. What's the difference between a destructor and a deconstruct method?
9. What are the signatures of the constructors that all exceptions should have?
10. What is an extension method, and how do you define one?

Exercise 6.2 – Practice creating an inheritance hierarchy

Explore inheritance hierarchies by following these steps:

1. Add a new console app named `Ch06Ex02Inheritance` to your `Chapter06` solution/workspace.
2. Create a class named `Shape` with properties named `Height`, `Width`, and `Area`.
3. Add three classes that derive from it— `Rectangle`, `Square`, and `Circle` —with any additional members you feel are appropriate and that override and implement the `Area` property correctly.
4. In `Program.cs`, add statements to create one instance of each shape, as shown in the following code:

```
Rectangle r = new(height: 3, width: 4.5);
WriteLine($"Rectangle H: {r.Height}, W: {r.Width}, Area: {r.Area}");
Square s = new(5);
WriteLine($"Square H: {s.Height}, W: {s.Width}, Area: {s.Area}");
Circle c = new(radius: 2.5);
WriteLine($"Circle H: {c.Height}, W: {c.Width}, Area: {c.Area}");
```

1. Run the console app and ensure that the result looks like the following output:

```
Rectangle H: 3, W: 4.5, Area: 13.5
Square H: 5, W: 5, Area: 25
Circle H: 5, W: 5, Area: 19.6349540849362
```

Exercise 6.3 – Explore topics

Use the links on the following page to learn more about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-6---implementing-interfaces-and-inheriting-classes

Summary

In this chapter, you learned about:

- Generic types.
- Delegates and events.
- Implementing interfaces.
- Memory usage differences between reference and value types.
- Working with null values.
- Deriving types using inheritance.
- Base and derived classes, how to override a type member, and using polymorphism.
- Casting between types.

In the next chapter, you will learn how .NET is packaged and deployed, and, in subsequent chapters, the types that it provides you with to implement common functionality such as file handling and database access.

7 Packaging and Distributing .NET Types

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about how C# keywords are related to .NET types, and about the relationship between namespaces and assemblies. You'll also become familiar with how to package and publish your .NET apps and libraries for cross-platform use, how to use legacy .NET Framework libraries in .NET libraries, and the possibility of porting legacy .NET Framework code bases to modern .NET.

This chapter covers the following topics:

- The road to .NET 7
- Understanding .NET components
- Publishing your applications for deployment
- Decompiling .NET assemblies
- Packaging your libraries for NuGet distribution
- Porting from .NET Framework to modern .NET
- Working with preview features

The road to .NET 7

This part of the book is about the functionality in the **Base Class Library** (**BCL**) APIs provided by .NET and how to reuse functionality across all the different .NET platforms using .NET Standard.

First, we will review the route to this point and why it is important to understand the past.

.NET Core 2.0 and later's support for a minimum of .NET Standard 2.0 is important because it provides many of the APIs that were missing from the first version of .NET Core. The 15 years' worth of libraries and applications that .NET Framework developers had available to them that are relevant for modern development have now been migrated to .NET and can run cross-platform on macOS and Linux variants, as well as on Windows.

.NET Standard 2.1 added about 3,000 new APIs. Some of those APIs need runtime changes that would break backward compatibility, so .NET Framework 4.8 only implements .NET Standard 2.0. .NET Core 3.0, Xamarin, Mono, and Unity implement .NET Standard 2.1.

.NET 5 removed the need for .NET Standard if all your projects could use .NET 5. The same applies to .NET 6 and .NET 7. Since you might still need to create class libraries for legacy .NET Framework projects or legacy Xamarin mobile apps, there is still a need to create .NET Standard 2.0 and 2.1 class libraries. In March 2021, I surveyed professional developers, and half still needed to create .NET Standard 2.0 compliant class libraries.

Now that .NET 7 has been released with full support for mobile and desktop apps built using .NET MAUI, the need for .NET Standard has been further reduced.

To summarize the progress that .NET has made over the past five years, I have compared the major .NET Core and modern .NET versions with the equivalent .NET Framework versions in the following list:

- **.NET Core 1.x**: Much smaller API compared to .NET Framework 4.6.1, which was the current version in March 2016.
- **.NET Core 2.x**: Reached API parity with .NET Framework 4.7.1 for modern APIs because they both implement .NET Standard 2.0.
- **.NET Core 3.x**: Larger API compared to .NET Framework for modern APIs because .NET Framework 4.8 does not implement .NET Standard 2.1.
- **.NET 5**: Even larger API compared to .NET Framework 4.8 for modern APIs, with much-improved performance.
- **.NET 6**: Continued improvements to performance and expanded APIs. Optional support for mobile apps in .NET MAUI added in May 2022.
- **.NET 7**: Final unification with the support for mobile apps in .NET MAUI.

.NET Core 1.0

.NET Core 1.0 was released in June 2016 and focused on implementing an API suitable for building modern cross-platform apps, including web and cloud applications and services for Linux using ASP.NET Core.

.NET Core 1.1

.NET Core 1.1 was released in November 2016 and focused on fixing bugs, increasing the number of Linux distributions supported, supporting .NET Standard 1.6, and improving performance, especially with ASP.NET Core for web apps and services.

.NET Core 2.0

.NET Core 2.0 was released in August 2017 and focused on implementing .NET Standard 2.0, the ability to reference .NET Framework libraries, and more performance improvements.

The third edition of this book was published in November 2017, so it covered up to .NET Core 2.0 and .NET Core for **Universal Windows Platform** (**UWP**) apps.

.NET Core 2.1

.NET Core 2.1 was released in May 2018 and focused on an extendable tooling system, adding new types like `Span<T>`, new APIs for cryptography and compression, a Windows Compatibility Pack with an additional 20,000 APIs to help port old Windows applications, Entity Framework Core value conversions, LINQ `GroupBy` conversions, data seeding, query types, and even more performance improvements, including the topics listed in the following table:

Feature	Chapter	Topic
Spans	8	Working with spans, indexes, and ranges
Brotli compression	9	Compressing with the Brotli algorithm
EF Core Lazy loading	10	Enabling lazy loading
EF Core Data seeding	10	Understanding data seeding

.NET Core 2.2

.NET Core 2.2 was released in December 2018 and focused on diagnostic improvements for the runtime, optional tiered compilation, and adding new features to ASP.NET Core and Entity Framework Core like spatial data support using types from the **NetTopologySuite** (**NTS**) library, query tags, and collections of owned entities.

.NET Core 3.0

.NET Core 3.0 was released in September 2019 and focused on adding support for building Windows desktop applications using Windows Forms (2001), **Windows Presentation Foundation** (**WPF**; 2006), and Entity Framework 6.3, side-by-side and app-local deployments, a fast JSON reader, serial port access and other pinout access for **Internet of Things** (**IoT**) solutions, and tiered compilation by default, including the topics listed in the following table:

Feature	Chapter	Topic
Embedding .NET in-app	7	Publishing your applications for deployment
Index and Range	8	Working with spans, indexes, and ranges
`System.Text.Json`	9	High-performance JSON processing

The fourth edition of this book was published in October 2019, so it covered some of the new APIs added in later versions up to .NET Core 3.0.

.NET Core 3.1

.NET Core 3.1 was released in December 2019 and focused on bug fixes and refinements so that it could be a **Long Term Support** (**LTS**) release, not losing support until December 2022.

.NET 5.0

.NET 5.0 was released in November 2020 and focused on unifying the various .NET platforms except mobile, refining the platform, and improving performance, including the topics listed in the following table:

Feature	Chapter	Topic
`Half` type	8	Working with numbers
Regular expression performance improvements	8	Regular expression performance improvements
`System.Text.Json` improvements	9	High-performance JSON processing
EF Core generated SQL	10	Getting the generated SQL
EF Core Filtered Include	10	Filtering included entities
EF Core Scaffold-DbContext now singularizes using Humanizer	10	Scaffolding models using an existing database

.NET 6.0

.NET 6.0 was released in November 2021 and focused on adding more features to EF Core for data management, new types for working with dates and times, and improving performance, including the topics listed in the following table:

Feature	Chapter	Topic
Check .NET SDK status	7	Checking your .NET SDKs for updates
Support for Apple Silicon	7	Creating a console application to publish
Link trim mode as default	7	Reducing the size of apps using app trimming
`DateOnly` and `TimeOnly`	8	Specifying date and time values
`EnsureCapacity` for `List<T>`	8	Improving performance by ensuring the capacity of a collection
Low-level file API using `RandomAccess`	9	Reading and writing with random access handles
EF Core configure conventions	10	Configuring preconvention models
New LINQ methods	11	Building LINQ expressions with the Enumerable class

.NET 7.0

.NET 7.0 was released in November 2022 and focused on unifying with the mobile platform, adding more features like x, and improving performance, including the topics listed in the following table:

Feature	Chapter	Topic
x		
x		
x		

Improving performance from .NET Core 2.0 to .NET 5 and later

Microsoft has made significant improvements to performance in the past few years. You can read detailed blog posts at the following links:

https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-5/

https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-6/

https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-7/

Checking your .NET SDKs for updates

With .NET 6, Microsoft added a command to check the versions of .NET SDKs and runtimes that you have installed and warn you if any need updating. For example, enter the following command:

```
dotnet sdk check
```

You will see results, including the status of available updates, as shown in the following partial output:

```
.NET SDKs:
Version                          Status
---------------------------------------------------------------------------
3.1.421                          .NET Core 3.1 is going out of support soon.
5.0.406                          .NET 5.0 is out of support.
6.0.300                          Patch 6.0.301 is available.
7.0.100                          Up to date.
```

Understanding .NET components

.NET is made up of several pieces, which are shown in the following list:

- **Language compilers**: These turn your source code written with languages such as C#, F#, and Visual Basic into **intermediate language** (**IL**) code stored in assemblies. With C# 6.0 and later, Microsoft switched to an open-source rewritten compiler known as Roslyn that is also used by Visual Basic.
- **Common Language Runtime (CoreCLR)**: This runtime loads assemblies, compiles the IL code stored in them into native code instructions for your computer's CPU, and executes the code within an environment that manages resources such as threads and memory.
- **Base Class Libraries (BCL or CoreFX)**: These are prebuilt assemblies of types packaged and distributed using NuGet for performing common tasks when building applications. You can use them to quickly build anything you want, rather like combining LEGO™ pieces.

Understanding assemblies, NuGet packages, and namespaces

An **assembly** is where a type is stored in the filesystem. Assemblies are a mechanism for deploying code. For example, the `System.Data.dll` assembly contains types for managing data. To use types in other assemblies, they must be referenced. Assemblies can be static (pre-created) or dynamic (generated at runtime). Dynamic assemblies are an advanced feature that we will not cover in this book. Assemblies can be compiled into a single file as a DLL (class library) or an EXE (console app).

Assemblies are distributed as **NuGet packages**, which are files downloadable from public online feeds and can contain multiple assemblies and other resources. You will also hear about **project SDKs**, **workloads**, and **platforms**, which are combinations of NuGet packages.

Microsoft's NuGet feed is found here: https://www.nuget.org/.

What is a namespace?

A namespace is the address of a type. Namespaces are a mechanism to uniquely identify a type by requiring a full address rather than just a short name. In the real world, *Bob of 34 Sycamore Street* is different from *Bob of 12 Willow Drive*.

In .NET, the `IActionFilter` interface of the `System.Web.Mvc` namespace is different from the `IActionFilter` interface of the `System.Web.Http.Filters` namespace.

Understanding dependent assemblies

If an assembly is compiled as a class library and provides types for other assemblies to use, then it has the file extension `.dll` (**dynamic link library**), and it cannot be executed standalone.

Likewise, if an assembly is compiled as an application, then it has the file extension `.exe` (**executable**) and can be executed standalone. Before .NET Core 3.0, console apps were compiled to `.dll` files and had to be executed by the `dotnet run` command or a host executable.

Any assembly can reference one or more class library assemblies as dependencies, but you cannot have circular references. So, assembly *B* cannot reference assembly *A* if assembly *A* already references assembly *B*. The compiler will warn you if you attempt to add a dependency reference that would cause a circular reference. Circular references are often a warning sign of poor code design. If you are sure that you need a circular reference, then use an interface to solve it.

Understanding the Microsoft .NET project SDKs

By default, console applications have a dependency reference on the Microsoft .NET project SDK. This platform contains thousands of types in NuGet packages that almost all applications would need, such as the `System.Int32` and `System.String` types.

When using .NET, you reference the dependency assemblies, NuGet packages, and platforms that your application needs in a project file.

Let's explore the relationship between assemblies and namespaces:

1. Use your preferred code editor to create a new project, as defined in the following list:
   - Project template: **Console App** / `console`
   - Project file and folder: `AssembliesAndNamespaces`
   - Workspace/solution file and folder: `Chapter07`
2. Open `AssembliesAndNamespaces.csproj` and note that it is a typical project file for a .NET application, as shown in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
</Project>
```

1. After the `<PropertyGroup>` section, add a new `<ItemGroup>` section to statically import `System.Console` for all C# files using the implicit usings .NET SDK feature, as shown in the following markup:

```
<ItemGroup>
  <Using Include="System.Console" Static="true" />
</ItemGroup>
```

Understanding namespaces and types in assemblies

Many common .NET types are in the `System.Runtime.dll` assembly. There is not always a one-to-one mapping between assemblies and namespaces. A single assembly can contain many namespaces and a namespace can be defined in many assemblies. You can see the relationship between some assemblies and the namespaces that they supply types for, as shown in the following table:

Assembly	Example namespaces	Example types
`System.Runtime.dll`	`System` , `System.Collections` , `System.Collections.Generic`	`Int32` , `String` , `IEnumerable<T>`
`System.Console.dll`	`System`	`Console`
`System.Threading.dll`	`System.Threading`	`Interlocked` , `Monitor` , `Mutex`
`System.Xml.XDocument.dll`	`System.Xml.Linq`	`XDocument` , `XElement` , `XNode`

Understanding NuGet packages

.NET is split into a set of packages, distributed using a Microsoft-supported package management technology named NuGet. Each of these packages represents a single assembly of the same name. For example, the `System.Collections` package contains the `System.Collections.dll` assembly.

The following are the benefits of packages:

- Packages can be easily distributed on public feeds.
- Packages can be reused.
- Packages can ship on their own schedule.
- Packages can be tested independently of other packages.
- Packages can support different OSes and CPUs by including multiple versions of the same assembly built for different OSes and CPUs.
- Packages can have dependencies specific to only one library.
- Apps are smaller because unreferenced packages aren't part of the distribution. The following table lists some of the more important packages and their important types:

Package	Important types
`System.Runtime`	`Object` , `String` , `Int32` , `Array`
`System.Collections`	`List<T>` , `Dictionary<TKey, TValue>`
`System.Net.Http`	`HttpClient` , `HttpResponseMessage`
`System.IO.FileSystem`	`File` , `Directory`
`System.Reflection`	`Assembly` , `TypeInfo` , `MethodInfo`

Understanding frameworks

There is a two-way relationship between frameworks and packages. Packages define the APIs, while frameworks group packages. A framework without any packages would not define any APIs.

.NET packages each support a set of frameworks. For example, the `System.IO.FileSystem` package version 4.3.0 supports the following frameworks:

- .NET Standard, version 1.3 or later.
- .NET Framework, version 4.6 or later.
- Six Mono and Xamarin platforms (for example, Xamarin.iOS 1.0).

More Information: You can read the details at the following link: https://www.nuget.org/packages/System.IO.FileSystem/.

Importing a namespace to use a type

Let's explore how namespaces are related to assemblies and types:

1. In the `AssembliesAndNamespaces` project, in `Program.cs`, delete the existing statements and then enter the following code:

```
XDocument doc = new();
```

1. Build the project and note the compiler error message, as shown in the following output:

```
The type or namespace name 'XDocument' could not be found (are you missing a using directive or a
```

The `XDocument` type is not recognized because we have not told the compiler what the namespace of the type is. Although this project already has a reference to the assembly that contains the type, we also need to either prefix the type name with its namespace or import the namespace.

1. Click inside the `XDocument` class name. Your code editor displays a light bulb, showing that it recognizes the type and can automatically fix the problem for you.
2. Click the light bulb, and select `using System.Xml.Linq;` from the menu.

This will *import the namespace* by adding a `using` statement to the top of the file. Once a namespace is imported at the top of a code file, then all the types within the namespace are available for use in that code file by just typing their name without the type name needing to be fully qualified by prefixing it with its namespace.

Sometimes I like to add a comment with a type name after importing a namespace to remind me why I need to import that namespace, as shown in the following code:

```
using System.Xml.Linq; // XDocument
```

Relating C# keywords to .NET types

One of the common questions I get from new C# programmers is, "What is the difference between `string` with a lowercase s and `String` with an uppercase S?"

The short answer is easy: none. The long answer is that all C# type keywords like `string` or `int` are aliases for a .NET type in a class library assembly.

When you use the `string` keyword, the compiler recognizes it as a `System.String` type. When you use the `int` type, the compiler recognizes it as a `System.Int32` type.

Let's see this in action with some code:

1. In `Program.cs`, declare two variables to hold `string` values, one using lowercase `string` and one using uppercase `String`, as shown in the following code:

```
string s1 = "Hello";
String s2 = "World";
WriteLine($"{s1} {s2}");
```

1. Run the code, and note that at the moment, they both work equally well, and literally mean the same thing.

2. In `AssembliesAndNamespaces.csproj`, add entries to prevent the `System` namespace from being globally imported, as shown in the following markup:

```
<ItemGroup>
  <Using Remove="System" />
</ItemGroup>
```

1. In `Program.cs` note the compiler error message, as shown in the following output:

```
The type or namespace name 'String' could not be found (are you missing a using directive or an a
```

1. At the top of `Program.cs`, import the `System` namespace with a `using` statement that will fix the error, as shown in the following code:

```
using System; // String
```

Good Practice: When you have a choice, use the C# keyword instead of the actual type because the keywords do not need the namespace imported.

Mapping C# aliases to .NET types

The following table shows the 18 C# type keywords along with their actual .NET types:

Keyword	.NET type	Keyword	.NET type
`string`	`System.String`	`char`	`System.Char`
`sbyte`	`System.SByte`	`byte`	`System.Byte`
`short`	`System.Int16`	`ushort`	`System.UInt16`
`int`	`System.Int32`	`uint`	`System.UInt32`
`long`	`System.Int64`	`ulong`	`System.UInt64`
`nint`	`System.IntPtr`	`nuint`	`System.UIntPtr`
`float`	`System.Single`	`double`	`System.Double`
`decimal`	`System.Decimal`	`bool`	`System.Boolean`
`object`	`System.Object`	`dynamic`	`System.Dynamic.DynamicObject`

Other .NET programming language compilers can do the same thing. For example, the Visual Basic .NET language has a type named `Integer` that is its alias for `System.Int32`.

Understanding native-sized integers

C# 9 introduced `nint` and `nuint` keyword alias for **native-sized integers**, meaning that the storage size for the integer value is platform-specific. They store a 32-bit integer in a 32-bit process and `sizeof()` returns 4 bytes; they store a 64-bit integer in a 64-bit process and `sizeof()` returns 8 bytes. The aliases represent pointers to the integer value in memory, which is why their .NET names are `IntPtr` and `UIntPtr`. The actual storage type will be either `System.Int32` or `System.Int64` depending on the process.

In a 64-bit process, the following code:

```
WriteLine($"int.MaxValue = {int.MaxValue:N0}");
WriteLine($"nint.MaxValue = {nint.MaxValue:N0}");
```

produces this output:

```
int.MaxValue = 2,147,483,647
nint.MaxValue = 9,223,372,036,854,775,807
```

Revealing the location of a type

Code editors provide built-in documentation for .NET types. Let's explore:

1. Right-click inside `XDocument` and choose **Go to Definition**.

2. Navigate to the top of the code file and note the assembly filename is `System.Xml.XDocument.dll`, but the class is in the `System.Xml.Linq` namespace, as shown in *Figure 7.1*:

Figure 7.1: Assembly and namespace that contains the XDocument type

3. Close the **XDocument [from metadata]** tab.
4. Right-click inside `string` or `String` and choose **Go to Definition**.
5. Navigate to the top of the code file and note the assembly filename is `System.Runtime.dll` but the class is in the `System` namespace.

Your code editor is technically lying to you. If you remember when we wrote code in *Chapter 2*, *Speaking C#*, when we revealed the extent of the C# vocabulary, we discovered that the `System.Runtime.dll` assembly contains zero types.

What the `System.Runtime.dll` assembly does contain are type-forwarders. These are special types that appear to exist in an assembly but are implemented elsewhere. In this case, they are implemented deep inside the .NET runtime using highly optimized code.

Sharing code with legacy platforms using .NET Standard

Before .NET Standard, there were **Portable Class Libraries** (**PCLs**). With PCLs, you could create a library of code and explicitly specify which platforms you want the library to support, such as Xamarin, Silverlight, and Windows 8. Your library could then use the intersection of APIs that are supported by the specified platforms.

Microsoft realized that this is unsustainable, so they created .NET Standard—a single API that all future .NET platforms would support. There are older versions of .NET Standard, but .NET Standard 2.0 was an attempt to unify all important recent .NET platforms. .NET Standard 2.1 was released in late 2019 but only .NET Core 3.0 and that year's version of Xamarin support its new features. For the rest of this book, I will use the term .NET Standard to mean .NET Standard 2.0.

.NET Standard is like HTML5 in that they are both standards that a platform should support. Just as Google's Chrome browser and Microsoft's Edge browser implement the HTML5 standard, .NET Core, .NET Framework, and Xamarin all implement .NET Standard. If you want to create a library of types that will work across variants of legacy .NET, you can do so most easily with .NET Standard.

> **Good Practice**: Since many of the API additions in .NET Standard 2.1 required runtime changes, and .NET Framework is Microsoft's legacy platform that needs to remain as unchanging as possible, .NET Framework 4.8 remained on .NET Standard 2.0 rather than implementing .NET Standard 2.1. If you need to support .NET Framework customers, then you should create class libraries on .NET Standard 2.0 even though it is not the latest and does not support all the recent language and BCL new features.

Your choice of which .NET Standard version to target comes down to a balance between maximizing platform support and available functionality. A lower version supports more platforms but has a smaller set of APIs. A higher version supports fewer platforms but has a larger set of APIs. Generally, you should choose the lowest version that supports all the APIs that you need.

Understanding defaults for class libraries with different SDKs

When using the `dotnet` SDK tool to create a class library, it might be useful to know which target framework will be used by default, as shown in the following table:

SDK	Default target framework for new class libraries
.NET Core 3.1	`netstandard2.0`
.NET 6	`net6.0`
.NET 7	`net7.0`

Of course, just because a class library targets a specific version of .NET by default does not mean you cannot change it after creating a class library project using the default template.

You can manually set the target framework to a value that supports the projects that need to reference that library, as shown in the following table:

Class library target framework	Can be used by projects that target
`netstandard2.0`	.NET Framework 4.6.1 or later, .NET Core 2.0 or later, .NET 5.0 or later, Mono 5.4 or later, Xamarin.Android 8.0 or later, Xamarin.iOS 10.14 or later
`netstandard2.1`	.NET Core 3.0 or later, .NET 5.0 or later, Mono 6.4 or later, Xamarin.Android 10.0 or later, Xamarin.iOS 12.16 or later
`net6.0`	.NET 6.0 or later
`net7.0`	.NET 7.0 or later

> **Good Practice**: Always check the target framework of a class library and then manually change it to something more appropriate if necessary. Make a conscious decision about what it should be rather than accepting the default.

Creating a .NET Standard 2.0 class library

We will create a class library using .NET Standard 2.0 so that it can be used across all important .NET legacy platforms and cross-platform on Windows, macOS, and Linux operating systems, while also having access to a

wide set of .NET APIs:

1. Use your preferred code editor to add a new **Class Library** / `classlib` project named `SharedLibrary` that targets .NET Standard 2.0 to the `Chapter07` solution/workspace.
   - If you use Visual Studio 2022, when prompted for the **Target Framework**, select **.NET Standard 2.0**, and then set the startup project for the solution to the current selection.
   - If you use Visual Studio Code, include a switch to target .NET Standard 2.0, as shown in the following command, and then select `SharedLibrary` as the active OmniSharp project:

```
dotnet new classlib -f netstandard2.0
```

> **Good Practice**: If you need to create types that use new features in .NET 7.0, as well as types that only use .NET Standard 2.0 features, then you can create two separate class libraries: one targeting .NET Standard 2.0 and one targeting .NET 7.0. You will see this in action in *Chapter 10*, *Working with Data Using Entity Framework Core*.

An alternative to manually creating two class libraries is to create one that supports **multi-targeting**. If you would like me to add a section about multi-targeting to the next edition, please let me know. You can read about multi-targeting here: https://docs.microsoft.com/en-us/dotnet/standard/library-guidance/cross-platform-targeting#multi-targeting.

Controlling the .NET SDK

By default, executing `dotnet` commands uses the most recent installed .NET SDK. There may be times when you want to control which SDK is used.

For example, one reader of the fourth edition wanted their experience to match the book steps that use the .NET Core 3.1 SDK. But they had installed the .NET 5.0 SDK as well and that was being used by default. As described in the previous section, the behavior when creating new class libraries changed to target .NET 5.0 instead of .NET Standard 2.0, and that confused the reader.

You can control the .NET SDK used by default by using a `global.json` file. The `dotnet` command searches the current folder and ancestor folders for a `global.json` file.

You do not need to complete the following steps, but if you want to try and do not already have .NET 6.0 SDK installed then you can install it from the following link:

https://dotnet.microsoft.com/download/dotnet/6.0

1. Create a subdirectory/folder in the `Chapter07` folder named `ControlSDK`.
2. On Windows, start **Command Prompt** or **Windows Terminal**. On macOS, start **Terminal**. If you are using Visual Studio Code, then you can use the integrated terminal.
3. In the `ControlSDK` folder, at the command prompt or terminal, enter a command to list the installed .NET SDKs, as shown in the following command:

```
dotnet --list-sdks
```

1. Note the results and the version number of the latest .NET 6 SDK installed, as shown in the following output:

```
3.1.416 [C:\Program Files\dotnet\sdk]
6.0.200 [C:\Program Files\dotnet\sdk]
7.0.100 [C:\Program Files\dotnet\sdk]
```

1. Create a `global.json` file that forces the use of the latest .NET Core 6.0 SDK that you have installed (which might be later than mine), as shown in the following command:

```
dotnet new globaljson --sdk-version 6.0.200
```

1. Open the `global.json` file and review its contents, as shown in the following markup:

```
{
  "sdk": {
    "version": "6.0.200"
  }
}
```

1. In the `ControlSDK` folder, at the command prompt or terminal, enter a command to create a class library project, as shown in the following command:

```
dotnet new classlib
```

1. If you do not have the .NET 6.0 SDK installed then you will see an error, as shown in the following output:

```
Could not execute because the application was not found or a compatible .NET SDK is not installe
```

1. If you do have the .NET 6.0 SDK installed, then a class library project will be created that targets .NET 6.0 by default.

Publishing your code for deployment

If you write a novel and you want other people to read it, you must publish it.

Most developers write code for other developers to use in their own projects, or for users to run as an app. To do so, you must publish your code as packaged class libraries or executable applications.

There are three ways to publish and deploy a .NET application. They are:

1. **Framework-dependent deployment** (**FDD**)
2. **Framework-dependent executable** (**FDE**)
3. **Self-contained**

If you choose to deploy your application and its package dependencies, but not .NET itself, then you rely on .NET already being on the target computer. This works well for web applications deployed to a server because .NET and lots of other web applications are likely already on the server.

Framework-dependent deployment (FDD) means you deploy a DLL that must be executed by the `dotnet` command-line tool. Framework-dependent executables (FDE) means you deploy an EXE that can be run directly from the command line. Both require the appropriate version of the .NET runtime to be already installed on the system.

Sometimes, you want to be able to give someone a USB stick containing your application and know that it can execute on their computer. You want to perform a self-contained deployment. While the size of the deployment files will be larger, you'll know that it will work.

Creating a console app to publish

Let's explore how to publish a console app:

1. Use your preferred code editor to add a new **Console App** / `console` project named `DotNetEverywhere` to the `Chapter07` solution/workspace.
   - In Visual Studio Code, select `DotNetEverywhere` as the active OmniSharp project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.
2. Modify the project file to statically import the `System.Console` class in all C# files.
3. In `Program.cs`, delete the existing statements and then add a statement to output a message saying the console app can run everywhere and some information about the operating system, as shown in the following code:

```
WriteLine("I can run everywhere!");
WriteLine($"OS Version is {Environment.OSVersion}.");
```

```
if (OperatingSystem.IsMacOS())
{
  WriteLine("I am macOS.");
}
else if (OperatingSystem.IsWindowsVersionAtLeast(major: 10, build: 22000))
{
  WriteLine("I am Windows 11.");
}
else if (OperatingSystem.IsWindowsVersionAtLeast(major: 10))
{
  WriteLine("I am Windows 10.");
}
else
{
  WriteLine("I am some other mysterious OS.");
}
WriteLine("Press ENTER to stop me.");
ReadLine();
```

1. Run the console app and note the results when run on Windows 11, as shown in the following output:

```
I can run everywhere!
OS Version is Microsoft Windows NT 10.0.22000.0.
I am Windows 11.
Press ENTER to stop me.
```

1. Open `DotNetEverywhere.csproj` and add the runtime identifiers to target three operating systems inside the `<PropertyGroup>` element, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <RuntimeIdentifiers>
      win10-x64;osx-x64;osx.11.0-arm64;linux-x64;linux-arm64
    </RuntimeIdentifiers>
  </PropertyGroup>
</Project>
```

- The `win10-x64` RID value means Windows 10 or Windows Server 2016 64-bit. You could also use the `win10-arm64` RID value to deploy to a Microsoft Surface Pro X.
- The `osx-x64` RID value means macOS Sierra 10.12 or later. You can also specify version-specific RID values like `osx.10.15-x64` (Catalina), `osx.11.0-x64` (Big Sur on Intel), or `osx.11.0-arm64` (Big Sur on Apple Silicon).
- The `linux-x64` RID value means most desktop distributions of Linux like Ubuntu, CentOS, Debian, or Fedora. Use `linux-arm` for Raspbian or Raspberry Pi OS 32-bit. Use `linux-arm64` for a Raspberry Pi running Ubuntu 64-bit.

Understanding dotnet commands

When you install the .NET SDK, it includes a **command-line interface (CLI)** named `dotnet`.

Creating new projects

The .NET CLI has commands that work on the current folder to create a new project using templates:

1. On Windows, start **Command Prompt** or **Windows Terminal**. On macOS, start **Terminal**. If you are using Visual Studio Code, then you can use the integrated terminal.
2. Enter the `dotnet new --list` or `dotnet new -l` command to list your currently installed templates, as shown in *Figure 7.2*:

```
PS C:\Users\markj> dotnet new -l
Template Name                                  Short Name          Language   Tags
---------------------------------------------- ------------------- ---------- ---------------------
Console Application                            console             [C#],F#,VB Common/Console
Class Library                                  classlib            [C#],F#,VB Common/Library
WPF Application                                wpf                 [C#],VB    Common/WPF
WPF Class Library                              wpflib              [C#],VB    Common/WPF
WPF Custom Control Library                     wpfcustomcontrollib [C#],VB    Common/WPF
WPF User Control Library                       wpfusercontrollib   [C#],VB    Common/WPF
Windows Forms App                              winforms            [C#],VB    Common/WinForms
Windows Forms Control Library                  winformscontrollib  [C#],VB    Common/WinForms
Windows Forms Class Library                    winformslib         [C#],VB    Common/WinForms
Worker Service                                 worker              [C#],F#    Common/Worker/Web
MSTest Test Project                            mstest              [C#],F#,VB Test/MSTest
NUnit 3 Test Item                              nunit-test          [C#],F#,VB Test/NUnit
NUnit 3 Test Project                           nunit               [C#],F#,VB Test/NUnit
xUnit Test Project                             xunit               [C#],F#,VB Test/xUnit
Razor Component                                razorcomponent      [C#]       Web/ASP.NET
Razor Page                                     page                [C#]       Web/ASP.NET
MVC ViewImports                                viewimports         [C#]       Web/ASP.NET
MVC ViewStart                                  viewstart           [C#]       Web/ASP.NET
Blazor Server App                              blazorserver        [C#]       Web/Blazor
Blazor WebAssembly App                         blazorwasm          [C#]       Web/Blazor/WebAssembly
ASP.NET Core Empty                             web                 [C#],F#    Web/Empty
ASP.NET Core Web App (Model-View-Controller)   mvc                 [C#],F#    Web/MVC
ASP.NET Core Web App                           webapp              [C#]       Web/MVC/Razor Pages
ASP.NET Core with Angular                      angular             [C#]       Web/MVC/SPA
ASP.NET Core with React.js                     react               [C#]       Web/MVC/SPA
ASP.NET Core with React.js and Redux           reactredux          [C#]       Web/MVC/SPA
Razor Class Library                            razorclasslib       [C#]       Web/Razor/Library
ASP.NET Core Web API                           webapi              [C#],F#    Web/WebAPI
ASP.NET Core gRPC Service                      grpc                [C#]       Web/gRPC
dotnet gitignore file                          gitignore                      Config
global.json file                               globaljson                     Config
NuGet Config                                   nugetconfig                    Config
Dotnet local tool manifest file                tool-manifest                  Config
Web Config                                     webconfig                      Config
Solution File                                  sln                            Solution
Protocol Buffer File                           proto                          Web/gRPC
```

Figure 7.2: A list of installed dotnet new project templates

Most `dotnet` command-line switches have a long and a short version. For example, `--list` or `-l`. The short ones are quicker to type but more likely to be misinterpreted by you or other humans. Sometimes more typing is clearer.

Getting information about .NET and its environment

It is useful to see what .NET SDKs and runtimes are currently installed, alongside information about the operating system, as shown in the following command:

```
dotnet --info
```

Note the results, as shown in the following partial output:

```
.NET SDK (reflecting any global.json):
 Version:   7.0.100
 Commit:    129d2465c8
Runtime Environment:
 OS Name:     Windows
 OS Version:  10.0.22000
 OS Platform: Windows
 RID:         win10-x64
 Base Path:   C:\Program Files\dotnet\sdk\7.0.100\
Host (useful for support):
  Version: 7.0.0
  Commit:  405337939c
.NET SDKs installed:
  3.1.416 [C:\Program Files\dotnet\sdk]
  5.0.405 [C:\Program Files\dotnet\sdk]
  6.0.200 [C:\Program Files\dotnet\sdk]
```

```
  7.0.100 [C:\Program Files\dotnet\sdk]
.NET runtimes installed:
  Microsoft.AspNetCore.App 3.1.22 [...\dotnet\shared\Microsoft.AspNetCore.All]
...
```

Managing projects

The .NET CLI has the following commands that work on the project in the current folder, to manage the project:

- `dotnet help` : Shows the command line help.
- `dotnet new` : Create a new .NET project or file.
- `dotnet tool` : Install or manage tools that extend the .NET experience.
- `dotnet workload` : Manage optional workloads like .NET MAUI.
- `dotnet restore` : This downloads dependencies for the project.
- `dotnet build` : This builds, aka compiles, a .NET project.
- `dotnet build-server` : Interact with servers started by a build.
- `dotnet msbuild` : This runs MS Build Engine commands.
- `dotnet clean` : This removes the temporary outputs from a build.
- `dotnet test` : This builds and then runs unit tests for the project.
- `dotnet run` : This builds and then runs the project.
- `dotnet pack` : This creates a NuGet package for the project.
- `dotnet publish` : This builds and then publishes the project, either with dependencies or as a self-contained application.
- `dotnet add` : This adds a reference to a package or class library to the project.
- `dotnet remove` : This removes a reference to a package or class library from the project.
- `dotnet list` : This lists the package or class library references for the project.

Publishing a self-contained app

Now that you have seen some example `dotnet` tool commands, we can publish our cross-platform console app:

1. At the command line, make sure that you are in the `DotNetEverywhere` folder.
2. Enter a command to build and publish the release version of the console application for Windows 10, as shown in the following command:

```
dotnet publish -c Release -r win10-x64
```

1. Note the build engine restores any needed packages, compiles the project source code into an assembly DLL, and creates a `publish` folder, as shown in the following output:

```
Microsoft (R) Build Engine version 17.2.0+09bdfae16 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.
  Determining projects to restore...
  Restored C:\cs11dotnet7\Chapter07\DotNetEverywhere\DotNetEverywhere.csproj (in 46.89 sec).
  DotNetEverywhere -> C:\cs11dotnet7\Chapter07\DotNetEverywhere\bin\Release\net7.0\win10-x64\Dotl
  DotNetEverywhere -> C:\cs11dotnet7\Chapter07\DotNetEverywhere\bin\Release\net7.0\win10-x64\publ
```

1. Enter the following commands to build and publish the release versions for macOS and Linux variants:

```
dotnet publish -c Release -r osx-x64
dotnet publish -c Release -r osx.11.0-arm64
dotnet publish -c Release -r linux-x64
dotnet publish -c Release -r linux-arm64
```

Good Practice: You could automate these commands by using a scripting language like PowerShell and execute the script file on any operating system using the cross-platform PowerShell Core. Just create a file with the extension `.ps1` with the five commands in it. Then execute the file. Learn more about PowerShell at the following link: https://github.com/markjprice/cs11dotnet7/tree/main/docs/powershell

1. Open a macOS **Finder** window or Windows **File Explorer**, navigate to `DotNetEverywhere\bin\Release\net7.0`, and note the output folders for the five operating systems.
2. In the `win10-x64` folder, select the `publish` folder, and note all the supporting assemblies like `Microsoft.CSharp.dll`.

3. Select the `DotNetEverywhere` executable file, and note it is 146 KB, as shown in *Figure 7.3*:

Figure 7.3: The DotNetEverywhere executable file for Windows 10 64-bit

4. If you are on Windows, then double-click to execute the program and note the result, as shown in the following output:

```
I can run everywhere!
OS Version is Microsoft Windows NT 10.0.22000.0.
I am Windows 11.
Press ENTER to stop me.
```

1. Note that the total size of the `publish` folder and all its files is about 70 MB.
2. In the `osx.11.0-arm64` folder, select the `publish` folder, note all the supporting assemblies, and then select the `DotNetEverywhere` executable file. Note that the executable is 126 KB, and the `publish` folder is about 76 MB.

If you copy any of those `publish` folders to the appropriate operating system, the console app will run; this is because it is a self-contained deployable .NET application. For example, on macOS with Intel, as shown in the following output:

```
I can run everywhere!
OS Version is Unix 11.2.3
I am macOS.
Press ENTER to stop me.
```

This example used a console app, but you could just as easily create an ASP.NET Core website or web service, or a Windows Forms or WPF app. Of course, you can only deploy Windows desktop apps to Windows computers, not Linux or macOS.

Publishing a single-file app

To publish as a "single" file, you can specify flags when publishing. With .NET 5, single-file apps were primarily focused on Linux because there are limitations in both Windows and macOS that mean true single-file publishing is not technically possible. With .NET 6 or later, you can now create proper single-file apps on Windows.

If you can assume that .NET is already installed on the computer on which you want to run your app, then you can use the extra flags when you publish your app for release to say that it does not need to be self-contained and that you want to publish it as a single file (if possible), as shown in the following command (that must be entered on a single line):

```
dotnet publish -r win10-x64 -c Release --self-contained=false
/p:PublishSingleFile=true
```

This will generate two files: `DotNetEverywhere.exe` and `DotNetEverywhere.pdb`. The `.exe` is the executable. The `.pdb` file is a **program debug database** file that stores debugging information.

There is no `.exe` file extension for published applications on macOS, so if you use `osx-x64` in the command above, the filename will not have an extension.

If you prefer the `.pdb` file to be embedded in the `.exe` file, for example, to ensure it is deployed with its assembly, then add a `<DebugType>` element to the `<PropertyGroup>` element in your `.csproj` file and set it to `embedded`, as shown highlighted in the following markup:

```
<PropertyGroup>
  <OutputType>Exe</OutputType>
  <TargetFramework>net7.0</TargetFramework>
  <Nullable>enable</Nullable>
  <ImplicitUsings>enable</ImplicitUsings>
  <RuntimeIdentifiers>
    win10-x64;osx-x64;osx.11.0-arm64;linux-x64;linux-arm64
  </RuntimeIdentifiers>
  <DebugType>embedded</DebugType>
</PropertyGroup>
```

If you cannot assume that .NET is already installed on a computer, then although Linux also only generates the two files, expect the following additional files for Windows: `coreclr.dll`, `clrjit.dll`, `clrcompression.dll`, and `mscordaccore.dll`.

Let's see an example for Windows:

1. At the command line, enter the command to build the release version of the console app for Windows 10, as shown in the following command:

```
dotnet publish -c Release -r win10-x64 /p:PublishSingleFile=true
```

1. Navigate to the `DotNetEverywhere\bin\Release\net7.0\win10-x64\publish` folder, select the `DotNetEverywhere` executable file, and note the executable is now about 63 MB, and there is also a `.pdb` file that is 11 KB. The sizes on your system will vary.

Reducing the size of apps using app trimming

One of the problems with deploying a .NET app as a self-contained app is that the .NET libraries take up a lot of space. One of the biggest needs to reduce the size of Blazor WebAssembly components because all the .NET libraries need to be downloaded to the browser.

Luckily, you can reduce this size by not packaging unused assemblies with your deployments. Introduced with .NET Core 3.0, the app trimming system can identify the assemblies needed by your code and remove those that are not needed.

With .NET 5, the trimming went further by removing individual types, and even members like methods from within an assembly if they are not used. For example, with a Hello World console app, the `System.Console.dll` assembly is trimmed from 61.5 KB to 31.5 KB. For .NET 5, this is an experimental feature, so it is disabled by default.

With .NET 6, Microsoft added annotations to their libraries to indicate how they can be safely trimmed so the trimming of types and members was made the default. This is known as **link trim mode**.

The catch is how well the trimming identifies unused assemblies, types, and members. If your code is dynamic, perhaps using reflection, then it might not work correctly, so Microsoft also allows manual control.

Enabling assembly-level trimming

There are two ways to enable assembly-level trimming.

The first way is to add an element in the project file, as shown in the following markup:

```
<PublishTrimmed>true</PublishTrimmed>
```

The second way is to add a flag when publishing, as shown highlighted in the following command:

```
dotnet publish ... -p:PublishTrimmed=True
```

Enabling type-level and member-level trimming

There are two ways to enable type-level and member-level trimming.

The first way is to add two elements in the project file, as shown in the following markup:

```
<PublishTrimmed>true</PublishTrimmed>
<TrimMode>Link</TrimMode>
```

The second way is to add two flags when publishing, as shown highlighted in the following command:

```
dotnet publish ... -p:PublishTrimmed=True -p:TrimMode=Link
```

For .NET 6, link trim mode is the default, so you only need to specify the switch if you want to set an alternative trim mode like `copyused`, which means assembly-level trimming.

Decompiling .NET assemblies

One of the best ways to learn how to code for .NET is to see how professionals do it.

> **Good Practice**: You could decompile someone else's assemblies for non-learning purposes like copying their code for use in your own production library or application, but remember that you are viewing their intellectual property, so please respect that.

Decompiling using the ILSpy extension for Visual Studio 2022

For learning purposes, you can decompile any .NET assembly with a tool like ILSpy.

1. In Visual Studio 2022 for Windows, navigate to **Extensions** | **Manage Extensions**.
2. In the search box, enter `ilspy`.
3. For the **ILSpy 2022** extension, click **Download**.
4. Click **Close**.
5. Close Visual Studio to allow the extension to install.
6. Restart Visual Studio and reopen the `Chapter07` solution.
7. In **Solution Explorer**, right-click the **DotNetEverywhere** project and select **Open output in ILSpy**.
8. In ILSpy, expand **DotNetEverywhere (1.0.0.0, .NETCoreApp, v7.0)**.
9. Expand **{ }**.
10. Select **Program**.

11. Click **[+]** to expand the statements in the compiler-generated `Program` class and `<Main>$` method to reveal how interpolated strings work, as shown in *Figure 7.4*:

Figure 7.4: Revealing the <Main>$ method and how interpolated strings work using ILSpy

12. In ILSpy, navigate to **File** | **Open…**.
13. Navigate to the following folder:

```
cs11dotnet7/Chapter07/DotNetEverywhere/bin/Release/net7.0/linux-x64
```

1. Select the `System.IO.FileSystem.dll` assembly and click **Open**.
2. In the **Assemblies** tree, select the **System.IO.FileSystem** assembly, and then click the `Directory` class in the `TypeForwardedTo` entry, as shown in the following code:

```
[assembly: TypeForwardedTo(typeof(Directory))]
```

1. In the `Directory` class, click the **[+]** to expand the `GetParent` method, and note the good practice of checking the `path` parameter and throwing an `ArgumentNullException` if it is `null` or an `ArgumentException` if it is zero length, as shown in *Figure 7.5*:

Figure 7.5: Decompiled GetParent method of Directory class on Windows

2. Review the C# source code for the `GetParent` method, as shown in the following code, in preparation for reviewing the same code in **Intermediate Language (IL)**:

```
public static DirectoryInfo? GetParent(string path)
{
  if (path == null)
  {
    throw new ArgumentNullException("path");
  }
  if (path.Length == 0)
  {
    throw new ArgumentException(SR.Argument_PathEmpty, "path");
  }
  string fullPath = Path.GetFullPath(path);
  string directoryName = Path.GetDirectoryName(fullPath);
  if (directoryName == null)
  {
    return null;
  }
  return new DirectoryInfo(directoryName);
}
```

1. In the ILSpy toolbar, click the **Select language to decompile** dropdown and select **IL**, and then review the IL source code of the `GetParent` method, as shown in the following code:

```
.method public hidebysig static
    class System.IO.DirectoryInfo GetParent (
        string path
    ) cil managed
{
    .param [0]
        .custom instance void System.Runtime.CompilerServices.NullableAttribute::.ctor(uint8) =
            01 00 02 00 00
        )
    // Method begins at RVA 0x57c848
    // Header size: 12
    // Code size: 64 (0x40)
    .maxstack 2
    .locals (
        [0] string,
        [1] string
    )
    IL_0000: ldarg.0
    IL_0001: brtrue.s IL_000e
    IL_0003: ldstr "path"
    IL_0008: newobj instance void System.ArgumentNullException::.ctor(string)
    IL_000d: throw
    IL_000e: ldarg.0
    IL_000f: callvirt instance int32 System.String::get_Length()
    IL_0014: brtrue.s IL_0026
    IL_0016: call string System.SR::get_Argument_PathEmpty()
    IL_001b: ldstr "path"
    IL_0020: newobj instance void System.ArgumentException::.ctor(string, string)
    IL_0025: throw
    IL_0026: ldarg.0
    IL_0027: call string System.IO.Path::GetFullPath(string)
    IL_002c: stloc.0
    IL_002d: ldloc.0
    IL_002e: call string System.IO.Path::GetDirectoryName(string)
    IL_0033: stloc.1
    IL_0034: ldloc.1
    IL_0035: brtrue.s IL_0039
    IL_0037: ldnull
    IL_0038: ret
    IL_0039: ldloc.1
    IL_003a: newobj instance void System.IO.DirectoryInfo::.ctor(string)
    IL_003f: ret
} // end of method Directory::GetParent
```

Good Practice: The IL code is not especially useful unless you get very advanced with C# and .NET development, when knowing how the C# compiler translates your source code into IL code can be important. The much more useful edit windows contain the equivalent C# source code written by Microsoft experts. You can learn a lot of good practices from seeing how professionals implement types. For example, the `GetParent` method shows how to check arguments for null and other argument exceptions.

1. Close ILSpy.

You can learn how to use the ILSpy extension for Visual Studio Code at the following link: https://github.com/markjprice/cs11dotnet7/blob/main/docs/code-editors/vscode.md#decompiling-using-the-ilspy-extension-for-visual-studio-code

No, you cannot technically prevent decompilation

I sometimes get asked if there is a way to protect compiled code to prevent decompilation. The quick answer is no, and if you think about it, you'll see why this must be the case. You can make it harder using obfuscation tools like **Dotfuscator**, but ultimately you cannot completely prevent it.

All compiled applications contain instructions to the platform, operating system, and hardware on which it runs. Those instructions must be functionally the same as the original source code but are just harder for a human to read. Those instructions must be readable to execute your code; they therefore must be readable to be decompiled. If you protected your code from decompilation using some custom technique, then you would also prevent your code from running!

Virtual machines simulate hardware and so can capture all interaction between your running application and the software and hardware that it thinks it is running on.

If you could protect your code, then you would also prevent attaching to it with a debugger and stepping through it. If the compiled application has a `pdb` file, then you can attach a debugger and step through the statements line-by-line. Even without the `pdb` file, you can still attach a debugger and get some idea of how the code works.

This is true for all programming languages. Not just .NET languages like C#, Visual Basic, and F#, but also C, C++, Delphi, assembly language: all can be attached to for debugging or to be disassembled or decompiled. Some tools used by professionals are shown in the following table:

Type	Product	Description
Virtual Machine	VMware	Professionals like malware analysts always run software inside a VM.
Debugger	SoftICE	Runs underneath the operating system, usually in a VM.
Debugger	WinDbg	Useful for understanding Windows internals because it knows more about Windows data structures than other debuggers.
Disassembler	IDA Pro	Used by professional malware analysts.
Decompiler	HexRays	Decompiles C apps. Plugin for IDA Pro.
Decompiler	DeDe	Decompiles Delphi apps.
Decompiler	dotPeek	.NET decompiler from JetBrains.

Good Practice: Debugging, disassembling, and decompiling someone else's software is likely against its license agreement and illegal in many jurisdictions. Instead of trying to protect your intellectual property with a technical solution, the law is sometimes your only recourse.

Packaging your libraries for NuGet distribution

Before we learn how to create and package our own libraries, we will review how a project can use an existing package.

Referencing a NuGet package

Let's say that you want to add a package created by a third-party developer, for example, `Newtonsoft.Json`, a popular package for working with the JavaScript Object Notation (JSON) serialization format:

1. In the `AssembliesAndNamespaces` project, add a reference to the `Newtonsoft.Json` NuGet package, either using the GUI for Visual Studio 2022 or the `dotnet add package` command for Visual Studio Code.
2. Open the `AssembliesAndNamespaces.csproj` file and note that a package reference has been added, as shown in the following markup:

```
<ItemGroup>
  <PackageReference Include="newtonsoft.json" Version="13.0.1" />
</ItemGroup>
```

If you have a more recent version of the `newtonsoft.json` package, then it has been updated since this chapter was written.

Fixing dependencies

To consistently restore packages and write reliable code, it's important that you **fix dependencies**. Fixing dependencies means you are using the same family of packages released for a specific version of .NET, for example, SQLite for .NET 7.0, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference
      Include="Microsoft.EntityFrameworkCore.Sqlite"
      Version="7.0.0" />
  </ItemGroup>
</Project>
```

To fix dependencies, every package should have a single version with no additional qualifiers. Additional qualifiers include betas (`beta1`), release candidates (`rc4`), and wildcards (`*`).

Wildcards allow future versions to be automatically referenced and used because they always represent the most recent release. But wildcards are therefore dangerous because they could result in the use of future incompatible packages that break your code.

This can be worth the risk while writing a book where new preview versions are released every month and you do not want to keep updating the package references, as I did during 2022, and as shown in the following markup:

```
<PackageReference
  Include="Microsoft.EntityFrameworkCore.Sqlite"
  Version="7.0.0-preview.*" />
```

If you use the `dotnet add package` command, or Visual Studio's **Manage NuGet Packages**, then it will by default use the latest specific version of a package. But if you copy and paste configuration from a blog article or manually add a reference yourself, you might include wildcard qualifiers.

The following dependencies are examples of NuGet package references that are *not* fixed and therefore should be avoided unless you know the implications:

```
<PackageReference Include="System.Net.Http" Version="4.1.0-*" />
<PackageReference Include="Newtonsoft.Json" Version="13.0.2-beta1" />
```

Good Practice: Microsoft guarantees that if you fixed your dependencies to what ships with a specific version of .NET, for example, 6.0.0, those packages will all work together. Almost always fix your dependencies.

Packaging a library for NuGet

Now, let's package the `SharedLibrary` project that you created earlier:

1. In the `SharedLibrary` project, rename the `Class1.cs` file to `StringExtensions.cs`.
2. Modify its contents to provide some useful extension methods for validating various text values using regular expressions, remembering that we are targeting .NET Standard 2.0 so the compiler is C# 8.0 by default and therefore we use older syntax for namespaces and so on, as shown in the following code:

```
using System.Text.RegularExpressions;
namespace Packt.Shared
{
  public static class StringExtensions
  {
    public static bool IsValidXmlTag(this string input)
    {
      return Regex.IsMatch(input,
        @"^<([a-z]+)([^<]+)*(?:>(.*)<\/\1>|\s+\/>)$");
    }
    public static bool IsValidPassword(this string input)
    {
      // minimum of eight valid characters
      return Regex.IsMatch(input, "^[a-zA-Z0-9_-]{8,}$");
    }
    public static bool IsValidHex(this string input)
    {
      // three or six valid hex number characters
      return Regex.IsMatch(input,
        "^#?([a-fA-F0-9]{3}|[a-fA-F0-9]{6})$");
    }
  }
}
```

You will learn how to write regular expressions in *Chapter 8*, *Working with Common .NET Types*.

1. In `SharedLibrary.csproj`, modify its contents, as shown highlighted in the following markup, and note the following:
   - `PackageId` must be globally unique, so you must use a different value if you want to publish this NuGet package to the https://www.nuget.org/ public feed for others to reference and download.
   - `PackageLicenseExpression` must be a value from the following link: https://spdx.org/licenses/, or you could specify a custom license.
   - All the other elements are self-explanatory:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>netstandard2.0</TargetFramework>
    <GeneratePackageOnBuild>true</GeneratePackageOnBuild>
    <PackageId>Packt.CSdotnet.SharedLibrary</PackageId>
    <PackageVersion>7.0.0.0</PackageVersion>
    <Title>C# 11 and .NET 7 Shared Library</Title>
    <Authors>Mark J Price</Authors>
    <PackageLicenseExpression>
      MS-PL
    </PackageLicenseExpression>
    <PackageProjectUrl>
      https://github.com/markjprice/cs11dotnet7
    </PackageProjectUrl>
    <PackageIcon>packt-csdotnet-sharedlibrary.png</PackageIcon>
    <PackageRequireLicenseAcceptance>true</PackageRequireLicenseAcceptance>
    <PackageReleaseNotes>
      Example shared library packaged for NuGet.
    </PackageReleaseNotes>
```

```
    <Description>
      Three extension methods to validate a string value.
    </Description>
    <Copyright>
      Copyright © 2016-2022 Packt Publishing Limited
    </Copyright>
    <PackageTags>string extensions packt csharp dotnet</PackageTags>
  </PropertyGroup>
  <ItemGroup>
    <None Include="packt-csdotnet-sharedlibrary.png">
      <Pack>True</Pack>
      <PackagePath></PackagePath>
    </None>
  </ItemGroup>
</Project>
```

Good Practice: Configuration property values that are `true` or `false` values cannot have any whitespace, so the `<PackageRequireLicenseAcceptance>` entry cannot have a carriage return and indentation as shown in the preceding markup.

1. Download the icon file and save it in the `SharedLibrary` folder from the following link: https://github.com/markjprice/cs11dotnet7/blob/main/vs4win/Chapter07/SharedLibrary/packt-csdotnet-sharedlibrary.png.
2. Build the release assembly:
   - In Visual Studio 2022, select **Release** in the toolbar, and then navigate to **Build** | **Build SharedLibrary**.
   - In Visual Studio Code, in **Terminal**, enter `dotnet build -c Release`

If we had not set `<GeneratePackageOnBuild>` to `true` in the project file, then we would have to create a NuGet package manually using the following additional steps:

- In Visual Studio 2022, navigate to **Build** | **Pack SharedLibrary**.
- In Visual Studio Code, in **Terminal**, enter `dotnet pack -c Release`.

Publishing a package to a public NuGet feed

If you want everyone to be able to download and use your NuGet package, then you must upload it to a public NuGet feed like Microsoft's:

1. Start your favorite browser and navigate to the following link: https://www.nuget.org/packages/manage/upload.
2. You will need to sign up for and then sign in with a Microsoft account at https://www.nuget.org/ if you want to upload a NuGet package for other developers to reference as a dependency package.
3. Click the **Browse...** button and select the `.nupkg` file that was created by generating the NuGet package. The folder path should be `cs11dotnet7\Chapter07\SharedLibrary\bin\Release` and the file is named `Packt.CSdotnet.SharedLibrary.7.0.0.nupkg`.
4. Verify that the information you entered in the `SharedLibrary.csproj` file has been correctly filled in, and then click **Submit**.

5. Wait a few seconds, and you will see a success message showing that your package has been uploaded, as shown in *Figure 7.6*:

Figure 7.6: A NuGet package upload message

Good Practice: If you get an error, then review the project file for mistakes, or read more information about the `PackageReference` format at https://docs.microsoft.com/en-us/nuget/reference/msbuild-targets.

6. Click the **Frameworks** tab, and note that because we targeted .NET Standard 2.0, our class library can be used by every .NET platform, as shown in *Figure 7.7*:

Figure 7.7: .NET Standard 2.0 class library NuGet packages can be used by all .NET platforms

Publishing a package to a private NuGet feed

Organizations can host their own private NuGet feeds. This can be a handy way for many developer teams to share work. You can read more at the following link:

https://docs.microsoft.com/en-us/nuget/hosting-packages/overview

Exploring NuGet packages with a tool

A handy tool named **NuGet Package Explorer** for opening and reviewing more details about a NuGet package was created by Uno Platform. As well as being a website, it can be installed as a cross-platform app. Let's see what it can do:

1. Start your favorite browser and navigate to the following link: https://nuget.info.
2. In the search box, enter `Packt.CSdotnet.SharedLibrary`.

3. Select the package **v7.0.0** published by **Mark J Price** and then click the **Open** button.
4. In the **Contents** section, expand the **lib** folder and the **netstandard2.0** folder.

5. Select **SharedLibrary.dll**, and note the details, as shown in *Figure 7.8*:

Figure 7.8: Exploring my package using NuGet Package Explorer from Uno Platform

6. If you want to use this tool locally in the future, click the install button in your browser.
7. Close your browser.

Not all browsers support installing web apps like this. I recommend Chrome for testing and development.

Testing your class library package

You will now test your uploaded package by referencing it in the `AssembliesAndNamespaces` project:

1. In the `AssembliesAndNamespaces` project, add a reference to your (or my) package, as shown highlighted in the following markup:

```
<ItemGroup>
  <PackageReference Include="newtonsoft.json" Version="13.0.1" />
  <PackageReference Include="packt.csdotnet.sharedlibrary"
    Version="7.0.0" />
</ItemGroup>
```

1. Build the `AssembliesAndNamespaces` console app.
2. In `Program.cs`, import the `Packt.Shared` namespace.
3. In `Program.cs`, prompt the user to enter some `string` values, and then validate them using the extension methods in the package, as shown in the following code:

```
Write("Enter a color value in hex: ");
string? hex = ReadLine(); // or "00ffc8"
WriteLine("Is {0} a valid color value? {1}",
  arg0: hex, arg1: hex.IsValidHex());
Write("Enter a XML element: ");
string? xmlTag = ReadLine(); // or "<h1 class=\"<\" />"
WriteLine("Is {0} a valid XML element? {1}",
  arg0: xmlTag, arg1: xmlTag.IsValidXmlTag());
Write("Enter a password: ");
string? password = ReadLine(); // or "secretsauce"
WriteLine("Is {0} a valid password? {1}",
  arg0: password, arg1: password.IsValidPassword());
```

1. Run the console app, enter some values as prompted, and view the results, as shown in the following output:

```
Enter a color value in hex: 00ffc8
Is 00ffc8 a valid color value? True
Enter an XML element: <h1 class="<" />
Is <h1 class="<" /> a valid XML element? False
Enter a password: secretsauce
Is secretsauce a valid password? True
```

Porting from .NET Framework to modern .NET

If you are an existing .NET Framework developer, then you may have existing applications that you think you should port to modern .NET. But you should carefully consider if porting is the right choice for your code, because sometimes, the best choice is not to port.

For example, you might have a complex website project that runs on .NET Framework 4.8 but is only visited by a small number of users. If it works and handles the visitor traffic on minimal hardware, then potentially spending months porting it to a modern .NET platform could be a waste of time. But if the website currently requires many expensive Windows servers, then the cost of porting could eventually pay off if you can migrate to fewer, less costly Linux servers.

Could you port?

Modern .NET has great support for the following types of applications on Windows, macOS, and Linux, so they are good candidates for porting:

- **ASP.NET Core MVC** websites.
- **ASP.NET Core Web API** web services (REST/HTTP).
- **ASP.NET Core SignalR** services.
- **Console App** command-line interfaces.

Modern .NET has decent support for the following types of applications on Windows, so they are potential candidates for porting:

- **Windows Forms** applications.
- **Windows Presentation Foundation** (**WPF**) applications.

Modern .NET has good support for the following types of applications on cross-platform desktop and mobile devices:

- **Xamarin** apps for mobile iOS and Android.
- **.NET MAUI** for desktop Windows and macOS, or mobile iOS and Android.

Modern .NET does not support the following types of legacy Microsoft projects:

- **ASP.NET Web Forms** websites. These might be best reimplemented using **ASP.NET Core Razor Pages** or **Blazor**.
- **Windows Communication Foundation** (**WCF**) services (but there is an open-source project named **CoreWCF** that you might be able to use depending on requirements). WCF services might be better reimplemented using **ASP.NET Core gRPC** services.
- **Silverlight** applications. These might be best reimplemented using **.NET MAUI**.

Silverlight and ASP.NET Web Forms applications will never be able to be ported to modern .NET, but existing Windows Forms and WPF applications could be ported to .NET on Windows to benefit from the new APIs and faster performance.

Legacy ASP.NET MVC web applications and ASP.NET Web API web services currently on .NET Framework could be ported to modern .NET and then be hosted on Windows, Linux, or macOS.

Should you port?

Even if you *could* port, *should* you? What benefits do you gain? Some common benefits include the following:

- **Deployment to Linux, Docker, or Kubernetes for websites and web services**: These OSes are lightweight and cost-effective as website and web service platforms, especially when compared to the more costly Windows Server.
- **Removal of dependency on IIS and System.Web.dll**: Even if you continue to deploy to Windows Server, ASP.NET Core can be hosted on lightweight, higher-performance Kestrel (or other) web servers.
- **Command-line tools**: Tools that developers and administrators use to automate their tasks are often built as console applications. The ability to run a single tool cross-platform is very useful.

Differences between .NET Framework and modern .NET

There are three key differences, as shown in the following table:

Modern .NET	.NET Framework
Distributed as NuGet packages, so each application can be deployed with its own app-local copy of the version of .NET that it needs.	Distributed as a system-wide, shared set of assemblies (literally, in the Global Assembly Cache (GAC)).
Split into small, layered components, so a minimal deployment can be performed.	Single, monolithic deployment.
Removes older technologies, such as ASP.NET Web Forms, and non-cross-platform features, such as AppDomains, .NET Remoting, and binary serialization.	As well as some similar technologies to those in modern .NET like ASP.NET Core MVC, it also retains some older technologies, such as ASP.NET Web Forms.

Understanding the .NET Portability Analyzer

Microsoft has a useful tool that you can run against your existing applications to generate a report for porting. You can watch a demonstration of the tool at the following link: https://docs.microsoft.com/en-us/shows/seth-juarez/brief-look-net-portability-analyzer.

Understanding the .NET Upgrade Assistant

Microsoft's latest tool for upgrading legacy projects to modern .NET is the .NET Upgrade Assistant.

For my day job, I used to work for a company named Optimizely. They have an enterprise-scale Digital Experience Platform (DXP) based on .NET comprising a Content Management System (CMS) and a Digital

Commerce platform. Microsoft needed a challenging migration project to design and test the .NET Upgrade Assistant with, so we worked with them to build a great tool.

Currently, it supports the following .NET Framework project types and more will be added later:

- ASP.NET MVC
- Windows Forms
- WPF
- Console Application
- Class Library

It is installed as a global `dotnet` tool, as shown in the following command:

```
dotnet tool install -g upgrade-assistant
```

You can read more about this tool and how to use it at the following link:

https://docs.microsoft.com/en-us/dotnet/core/porting/upgrade-assistant-overview

Using non-.NET Standard libraries

Most existing NuGet packages can be used with modern .NET, even if they are not compiled for .NET Standard or a modern version like .NET 7. If you find a package that does not officially support .NET Standard, as shown on its https://nuget.org web page, you do not have to give up. You should try it and see if it works.

For example, there is a package of custom collections for handling matrices created by Dialect Software LLC, documented at the following link:

https://www.nuget.org/packages/DialectSoftware.Collections.Matrix/

This package was last updated in 2013, which was long before .NET Core or .NET 7 existed, so this package was built for .NET Framework. If an assembly package like this only uses APIs available in .NET Standard, it can be used in a modern .NET project.

Let's try using it and see if it works:

1. In the `AssembliesAndNamespaces` project, add a package reference for Dialect Software's package, as shown in the following markup:

```
<PackageReference
  Include="dialectsoftware.collections.matrix"
  Version="1.0.0" />
```

1. Build the `AssembliesAndNamespaces` project to restore packages.
2. In `Program.cs`, add statements to import the `DialectSoftware.Collections` and `DialectSoftware.Collections.Generics` namespaces.
3. Add statements to create instances of `Axis` and `Matrix<T>`, populate them with values, and output them, as shown in the following code:

```
Axis x = new("x", 0, 10, 1);
Axis y = new("y", 0, 4, 1);
Matrix<long> matrix = new(new[] { x, y });
for (int i = 0; i < matrix.Axes[0].Points.Length; i++)
{
  matrix.Axes[0].Points[i].Label = "x" + i.ToString();
}
for (int i = 0; i < matrix.Axes[1].Points.Length; i++)
{
  matrix.Axes[1].Points[i].Label = "y" + i.ToString();
}
foreach (long[] c in matrix)
```

```
{
  matrix[c] = c[0] + c[1];
}
foreach (long[] c in matrix)
{
  WriteLine("{0},{1} ({2},{3}) = {4}",
    matrix.Axes[0].Points[c[0]].Label,
    matrix.Axes[1].Points[c[1]].Label,
    c[0], c[1], matrix[c]);
}
```

1. Run the code, noting the warning message and the results, as shown in the following output:

```
warning NU1701: Package 'DialectSoftware.Collections.Matrix
1.0.0' was restored using '.NETFramework,Version=v4.6.1,
.NETFramework,Version=v4.6.2, .NETFramework,Version=v4.7,
.NETFramework,Version=v4.7.1, .NETFramework,Version=v4.7.2,
.NETFramework,Version=v4.8' instead of the project target framework 'net7.0'. This package may n
x0,y0 (0,0) = 0
x0,y1 (0,1) = 1
x0,y2 (0,2) = 2
x0,y3 (0,3) = 3
...
```

Even though this package was created before modern .NET existed, and the compiler and runtime have no way of knowing if it will work and therefore show warnings, because it happens to only call .NET Standard-compatible APIs, it works.

Working with preview features

It is a challenge for Microsoft to deliver some new features that have cross-cutting effects across many parts of .NET like the runtime, language compilers, and API libraries. It is the classic chicken and egg problem. What do you do first?

From a practical perspective, it means that although Microsoft might have completed most of the work needed for a feature, the whole thing might not be ready until very late in their now annual cycle of .NET releases, too late for proper testing in "the wild."

So, from .NET 6 onward, Microsoft will include preview features in **general availability** (**GA**) releases. Developers can opt into these preview features and provide Microsoft with feedback. In a later GA release, they can be enabled for everyone.

For example, when Microsoft released .NET SDK 6.0.200 in February 2022, it included the C# 11 compiler as a preview feature. This meant that .NET 6 developers could optionally set language version to `preview`, and then start exploring C# 11 features like `!!` to automatically add `null` parameter checks.

> **Good Practice**: Preview features are not supported in production code. Preview features are likely to have breaking changes before the final release. Enable preview features at your own risk.

Requiring preview features

The `[RequiresPreviewFeatures]` attribute is used to indicate assemblies, types, or members that use and therefore require warnings about preview features. A code analyzer then scans for this assembly and generates warnings if needed. If your code does not use any preview features, you will not see any warnings. If you use any preview features, then your code should warn consumers of your code that you use preview features.

Enabling preview features

Let's look at an example of a preview feature available in .NET 7, ?:

1. Use your preferred code editor to add a new **Console App** / `console` project named `UsingPreviewFeatures` to the `Chapter07` solution/workspace.
    - In Visual Studio Code, select `UsingPreviewFeatures` as the active OmniSharp project. When you see the pop-up warning message saying that required assets are missing, click **Yes** to add them.
2. In the project file, add an element to enable preview features and an element to enable preview language features, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <EnablePreviewFeatures>true</EnablePreviewFeatures>
    <LangVersion>preview</LangVersion>
  </PropertyGroup>
</Project>
```

1. In `Program.cs`, delete the existing statements and then add statements to ?, as shown in the following code:

```
Doer.DoSomething();
public interface IWithStaticAbstract
{
  static abstract void DoSomething();
}
public class Doer : IWithStaticAbstract
{
  public static void DoSomething()
  {
    WriteLine("I am an implementation of a static abstract method.");
  }
}
```

1. Run the console app and note that it outputs correctly.

Generic mathematics

Why has Microsoft added the ability to define static abstract methods? What are they useful for?

For a long time, developers have asked Microsoft for the ability to use operators like * on generic types. This would enable a developer to define mathematical methods to perform operations like adding, averaging, and so on to any generic type rather than having to create dozens of overloaded methods for all the numeric types they want to support. Support for static abstract methods in interfaces is a foundational feature that would enable generic mathematics.

If you are interested, you can read more about this at the following link:

https://devblogs.microsoft.com/dotnet/preview-features-in-net-6-generic-math/

Practicing and exploring

Test your knowledge and understanding by answering some questions, getting some hands-on practice, and exploring with deeper research into topics of this chapter.

Exercise 7.1 – Test your knowledge

Answer the following questions:

1. What is the difference between a namespace and an assembly?
2. How do you reference another project in a `.csproj` file?
3. What is the benefit of a tool like ILSpy?

4. Which .NET type does the C# `float` alias represent?
5. When porting an application from .NET Framework to .NET 6, what tool should you run before porting, and what tool could you run to perform much of the porting work?
6. What is the difference between framework-dependent and self-contained deployments of .NET applications?
7. What is a RID?
8. What is the difference between the `dotnet pack` and `dotnet publish` commands?
9. What types of applications written for the .NET Framework can be ported to modern .NET?
10. Can you use packages written for .NET Framework with modern .NET?

Exercise 7.2 – Explore topics

Use the links on the following page to learn more detail about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-7---packaging-and-distributing-net-types

Exercise 7.3 – Explore PowerShell

PowerShell is Microsoft's scripting language for automating tasks on every operating system. Microsoft recommends Visual Studio Code with the PowerShell extension for writing PowerShell scripts.

Since PowerShell is its own extensive language, there is not space in this book to cover it. Instead, I have created some supplementary pages on the book's GitHub repository to introduce you to some key concepts and show some examples:

https://github.com/markjprice/cs11dotnet7/tree/main/docs/powershell

Summary

In this chapter, we:

- Reviewed the journey to .NET 7 for Base Class Library functionality.
- Explored the relationship between assemblies and namespaces.
- Learned how to decompile .NET assemblies for education purposes.
- Saw options for publishing an app for distribution to multiple operating systems.
- Packaged and distributed a class library.
- Discussed options for porting existing .NET Framework code bases.
- Learned how to activate preview features.

In the next chapter, you will learn about some common Base Class Library types that are included with modern .NET.

8 Working with Common .NET Types

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about some common types that are included with .NET. These include types for manipulating numbers, text, collections, improving working with spans, indexes, and ranges, and network access.

This chapter covers the following topics:

- Working with numbers
- Working with text
- Pattern matching with regular expressions
- Storing multiple objects in collections
- Working with spans, indexes, and ranges
- Working with network resources

Working with numbers

One of the most common types of data is numbers. The most common types in .NET for working with numbers are shown in the following table:

Namespace	Example type(s)	Description
`System`	`SByte` , `Int16` , `Int32` , `Int64`	Integers; that is, zero, and positive and negative whole numbers
`System`	`Byte` , `UInt16` , `UInt32` , `UInt64`	Cardinals; that is, zero, and positive whole numbers
`System`	`Half` , `Single` , `Double`	Reals; that is, floating-point numbers
`System`	`Decimal`	Accurate reals; that is, for use in science, engineering, or financial scenarios
`System.Numerics`	`BigInteger` , `Complex` , `Quaternion`	Arbitrarily large integers, complex numbers, and quaternion numbers

.NET has had the 32-bit float and 64-bit double types since .NET Framework 1.0. The IEEE 754 specification also defines a 16-bit floating point standard. Machine learning and other algorithms would benefit from this smaller, lower-precision number type so Microsoft introduced the `System.Half` type with .NET 5 and later.

Currently, the C# language does not define a `half` alias so you must use the .NET type `System.Half`. This might change in the future.

Working with big integers

The largest whole number that can be stored in .NET types that have a C# alias is about eighteen and a half quintillion, stored in an unsigned `long` integer. But what if you need to store numbers larger than that?

Let's explore numerics:

1. Use your preferred code editor to create a new project, as defined in the following list:
   - Project template: **Console App** / `console`
   - Project file and folder: `WorkingWithNumbers`
   - Workspace/solution file and folder: `Chapter08`
2. In the project file, add an element to statically and globally import the `System.Console` class.
3. In `Program.cs`, delete the existing statements and then add a statement to import `System.Numerics`, as shown in the following code:

```
using System.Numerics;
```

1. Add statements to output the maximum value of the `ulong` type, and a number with 30 digits using `BigInteger`, as shown in the following code:

```
WriteLine("Working with large integers:");
WriteLine("------------------------------------");
ulong big = ulong.MaxValue;
WriteLine($"{big,40:N0}");
BigInteger bigger =
  BigInteger.Parse("123456789012345678901234567890");
WriteLine($"{bigger,40:N0}");
```

The `40` in the format code means "right-align 40 characters", so both numbers are lined up to the right-hand edge. The `N0` means "use thousand separators and zero decimal places".

1. Run the code and view the result, as shown in the following output:

```
Working with large integers:
----------------------------------------
              18,446,744,073,709,551,615
 123,456,789,012,345,678,901,234,567,890
```

Working with complex numbers

A complex number can be expressed as $a + bi$, where a and b are real numbers, and i is an imaginary unit, where $i^2 = -1$. If the real part a is zero, it is a pure imaginary number. If the imaginary part b is zero, it is a real number.

Complex numbers have practical applications in many **STEM** (**science, technology, engineering, and mathematics**) fields of study. They are added by separately adding the real and imaginary parts of the summands; consider this:

```
(a + bi) + (c + di) = (a + c) + (b + d)i
```

Let's explore complex numbers:

1. In `Program.cs`, add statements to add two complex numbers, as shown in the following code:

```
WriteLine("Working with complex numbers:");
Complex c1 = new(real: 4, imaginary: 2);
Complex c2 = new(real: 3, imaginary: 7);
Complex c3 = c1 + c2;
// output using default ToString implementation
WriteLine($"{c1} added to {c2} is {c3}");
// output using custom format
```

```
WriteLine("{0} + {1}i added to {2} + {3}i is {4} + {5}i",
  c1.Real, c1.Imaginary,
  c2.Real, c2.Imaginary,
  c3.Real, c3.Imaginary);
```

1. Run the code and view the result, as shown in the following output:

```
Working with complex numbers:
(4, 2) added to (3, 7) is (7, 9)
4 + 2i added to 3 + 7i is 7 + 9i
```

Understanding quaternions

Quaternions are a number system that extends complex numbers. They form a four-dimensional associative normed division algebra over the real numbers, and therefore also a domain.

Huh? Yes, I know. I don't understand that either. Don't worry; we're not going to write any code using them! Suffice to say, they are good at describing spatial rotations, so video game engines use them, as do many computer simulations and flight control systems.

Generating random numbers for games and similar apps

In scenarios that don't need truly random numbers like games, you can create an instance of the `Random` class, as shown in the following code example:

```
Random r = new();
```

Random has a constructor with a parameter for specifying a seed value used to initialize its pseudo-random number generator, as shown in the following code:

```
Random r = new(Seed: 46378);
```

As you learned in *Chapter 2*, *Speaking C#*, parameter names should use camel case. The developer who defined the constructor for the `Random` class broke this convention! The parameter name should be `seed`, not `Seed`.

> **Good Practice**: Shared seed values act as a secret key, so if you use the same random number generation algorithm with the same seed value in two applications, then they can generate the same "random" sequences of numbers. Sometimes this is necessary, for example, when synchronizing a GPS receiver with a satellite, or when a game needs to randomly generate the same level. But usually, you want to keep your seed secret.

To avoid allocating more memory, .NET 6 introduced a shared static instance of `Random`, as shown in the following code:

```
Random r = Random.Shared;
```

Once you have a `Random` object, you can call its methods to generate random numbers, as shown in the following code examples:

```
// minValue is an inclusive lower bound i.e. 1 is a possible value
// maxValue is an exclusive upper bound i.e. 7 is not a possible value
int dieRoll = r.Next(minValue: 1, maxValue: 7); // returns 1 to 6
double randomReal = r.NextDouble(); // returns 0.0 to less than 1.0
byte[] arrayOfBytes = new byte[256];
r.NextBytes(arrayOfBytes); // 256 random bytes in an array
```

The `Next` method takes two parameters: `minValue` and `maxValue`. But `maxValue` is not the maximum value that the method returns! It is an *exclusive upper bound*, meaning it is one more than the maximum value. In a similar way, the value returned by the `NextDouble` method is greater than or equal to `0.0` and less than `1.0`. `NextBytes` populates an array of any size with random `byte` (0 to 255) values.

> In scenarios that do need truly random numbers like cryptography, there are specialized types for that, like `RandomNumberGenerator`. I cover this and other cryptographic types in the companion book, *Apps and Services with .NET 7*.

Working with text

One of the other most common types of data for variables is text. The most common types in .NET for working with text are shown in the following table:

Namespace	Type	Description
`System`	`Char`	Storage for a single text character
`System`	`String`	Storage for multiple text characters
`System.Text`	`StringBuilder`	Efficiently manipulates strings
`System.Text.RegularExpressions`	`Regex`	Efficiently pattern-matches strings

Getting the length of a string

Let's explore some common tasks when working with text; for example, sometimes you need to find out the length of a piece of text stored in a `string` variable:

1. Use your preferred code editor to add a new **Console App** / `console` project named `WorkingWithText` to the `Chapter08` solution/workspace:
   - In Visual Studio, set the startup project for the solution to the current selection.
   - In Visual Studio Code, select `WorkingWithText` as the active OmniSharp project.
2. In the `WorkingWithText` project, in `Program.cs`, delete the existing statements and then add statements to define a variable to store the name of the city London, and then write its name and length to the console, as shown in the following code:

```
string city = "London";
WriteLine($"{city} is {city.Length} characters long.");
```

1. Run the code and view the result, as shown in the following output:

```
London is 6 characters long.
```

Getting the characters of a string

The `string` class uses an array of `char` internally to store the text. It also has an indexer, which means that we can use the array syntax to read its characters. Array indexes start at zero, so the third character will be at index 2.

Let's see this in action:

1. Add a statement to write the characters at the first and fourth positions in the `string` variable, as shown in the following code:

```
WriteLine($"First char is {city[0]} and fourth is {city[3]}.");
```

1. Run the code and view the result, as shown in the following output:

```
First char is L and fourth is d.
```

Splitting a string

Sometimes, you need to split some text wherever there is a character, such as a comma:

1. Add statements to define a single `string` variable containing comma-separated city names, then use the `Split` method and specify that you want to treat commas as the separator, and then enumerate the returned array of `string` values, as shown in the following code:

```
string cities = "Paris,Tehran,Chennai,Sydney,New York,Medellín";
string[] citiesArray = cities.Split(',');
WriteLine($"There are {citiesArray.Length} items in the array:");
foreach (string item in citiesArray)
{
  WriteLine(item);
}
```

1. Run the code and view the result, as shown in the following output:

```
There are 6 items in the array:
Paris
Tehran
Chennai
Sydney
New York
Medellín
```

Later in this chapter, you will learn how to handle more complex scenarios.

Getting part of a string

Sometimes, you need to get part of some text. The `IndexOf` method has nine overloads that return the index position of a specified `char` or `string` within a `string`. The `Substring` method has two overloads, as shown in the following list:

- `Substring(startIndex, length)` : Returns part of a string starting at `startIndex` and containing the next `length` characters.
- `Substring(startIndex)` : Returns part of a string starting at `startIndex` and containing all characters up to the end of the string.

Let's explore a simple example:

1. Add statements to store a person's full name in a `string` variable with a space character between the first and last name, find the position of the space, and then extract the first name and last name as two parts so that they can be recombined in a different order, as shown in the following code:

```
string fullName = "Alan Shore";
int indexOfTheSpace = fullName.IndexOf(' ');
string firstName = fullName.Substring(
  startIndex: 0, length: indexOfTheSpace);
string lastName = fullName.Substring(
  startIndex: indexOfTheSpace + 1);
WriteLine($"Original: {fullName}");
WriteLine($"Swapped: {lastName}, {firstName}");
```

1. Run the code and view the result, as shown in the following output:

```
Original: Alan Shore
Swapped: Shore, Alan
```

If the format of the initial full name was different, for example, `"LastName, FirstName"`, then the code would need to be different. As an optional exercise, try writing some statements that would change the input `"Shore, Alan"` into `"Alan Shore"`.

Checking a string for content

Sometimes, you need to check whether a piece of text starts or ends with some characters or contains some characters. You can achieve this with methods named `StartsWith`, `EndsWith`, and `Contains`:

1. Add statements to store a `string` value and then check if it starts with or contains a couple of different `string` values, as shown in the following code:

```
string company = "Microsoft";
bool startsWithM = company.StartsWith("M");
bool containsN = company.Contains("N");
WriteLine($"Text: {company}");
WriteLine($"Starts with M: {startsWithM}, contains an N: {containsN}");
```

1. Run the code and view the result, as shown in the following output:

```
Text: Microsoft
Starts with M: True, contains an N: False
```

Joining, formatting, and other string members

There are many other `string` members, as shown in the following table:

Member	Description
`Trim, TrimStart, TrimEnd`	These methods trim whitespace characters such as space, tab, and carriage return from the beginning and/or end.
`ToUpper, ToLower`	These convert all the characters into uppercase or lowercase.
`Insert, Remove`	These methods insert or remove some text.
`Replace`	This replaces some text with other text.
`string.Empty`	This can be used instead of allocating memory each time you use a literal string value using an empty pair of double quotes ("").
`string.Concat`	This concatenates two string variables. The `+` operator does the equivalent when used between string operands.
`string.Join`	This concatenates one or more string variables with a character in between each one.
`string.IsNullOrEmpty`	This checks whether a string variable is null or empty.
`string.IsNullOrWhitespace`	This checks whether a string variable is null or whitespace; that is, a mix of any number of horizontal and vertical spacing characters, for example, tab, space, carriage return, line feed, and so on.
`string.Format`	An alternative method to string interpolation for outputting formatted string values, which uses positioned instead of named parameters.

Some of the preceding methods are `static` methods. This means that the method can only be called from the type, not from a variable instance. In the preceding table, I indicated the static methods by prefixing them with `string.`, as in `string.Format`.

Let's explore some of these methods:

1. Add statements to take an array of `string` values and combine them back together into a single `string` variable with separators using the `Join` method, as shown in the following code:

```
string recombined = string.Join(" => ", citiesArray);
WriteLine(recombined);
```

1. Run the code and view the result, as shown in the following output:

```
Paris => Tehran => Chennai => Sydney => New York => Medellín
```

1. Add statements to use positioned parameters and interpolated `string` formatting syntax to output the same three variables twice, as shown in the following code:

```
string fruit = "Apples";
decimal price =  0.39M;
DateTime when = DateTime.Today;
WriteLine($"Interpolated:  {fruit} cost {price:C} on {when:dddd}.");
WriteLine(string.Format("string.Format: {0} cost {1:C} on {2:dddd}.",
  arg0: fruit, arg1: price, arg2: when));
```

1. Run the code and view the result, as shown in the following output:

```
Interpolated:  Apples cost £0.39 on Thursday.
string.Format: Apples cost £0.39 on Thursday.
```

Note that we could have simplified the second statement because `Console.WriteLine` supports the same format codes as `string.Format`, as shown in the following code:

```
WriteLine("WriteLine: {0} cost {1:C} on {2:dddd}.",
  arg0: fruit, arg1: price, arg2: when);
```

Building strings efficiently

You can concatenate two strings to make a new `string` using the `String.Concat` method or simply by using the `+` operator. But both choices are bad practice because .NET must create a completely new `string` in memory.

This might not be noticeable if you are only adding two `string` values, but if you concatenate inside a loop with many iterations, it can have a significant negative impact on performance and memory use. You can concatenate `string` variables more efficiently using the `StringBuilder` type, which you can read more about at the following link:

https://docs.microsoft.com/en-us/dotnet/api/system.text.stringbuilder#examples

Pattern matching with regular expressions

Regular expressions are useful for validating input from the user. They are very powerful and can get very complicated. Almost all programming languages have support for regular expressions and use a common set of special characters to define them.

Let's try out some example regular expressions:

1. Use your preferred code editor to add a new **Console App** / console `project` named `WorkingWithRegularExpressions` to the `Chapter08` solution/workspace.
   - In Visual Studio Code, select `WorkingWithRegularExpressions` as the active OmniSharp project.
2. In `Program.cs`, delete the existing statements and then import the following namespace:

```
using System.Text.RegularExpressions; // Regex
```

Checking for digits entered as text

We will start by implementing the common example of validating number input:

1. In `Program.cs`, add statements to prompt the user to enter their age and then check that it is valid using a regular expression that looks for a digit character, as shown in the following code:

```
Write("Enter your age: ");
string input = ReadLine()!; // null-forgiving
Regex ageChecker = new(@"\d");
if (ageChecker.IsMatch(input))
{
  WriteLine("Thank you!");
}
else
{
  WriteLine($"This is not a valid age: {input}");
}
```

Note the following about the code:

- The `@` character switches off the ability to use escape characters in the `string`. Escape characters are prefixed with a backslash. For example, `\t` means a tab and `\n` means a new line. When writing regular expressions, we need to disable this feature. To paraphrase the television show The West Wing, "Let backslash be backslash."
- Once escape characters are disabled with `@`, then they can be interpreted by a regular expression. For example, `\d` means digit. You will learn more regular expressions that are prefixed with a backslash later in this topic.

1. Run the code, enter a whole number such as `34` for the age, and view the result, as shown in the following output:

```
Enter your age: 34
Thank you!
```

1. Run the code again, enter `carrots`, and view the result, as shown in the following output:

```
Enter your age: carrots
This is not a valid age: carrots
```

1. Run the code again, enter `bob30smith`, and view the result, as shown in the following output:

```
Enter your age: bob30smith
Thank you!
```

The regular expression we used is `\d`, which means *one digit*. However, it does not specify what can be entered before and after that one digit. This regular expression could be described in English as "Enter any characters you want as long as you enter at least one digit character."

In regular expressions, you indicate the start of some input with the caret `^` symbol and the end of some input with the dollar `$` symbol. Let's use these symbols to indicate that we expect nothing else between the start and end of the input except for a digit.

1. Change the regular expression to `^\d$`, as shown highlighted in the following code:

```
Regex ageChecker = new(@"^\d$");
```

1. Run the code again and note that it rejects any input except a single digit. We want to allow one or more digits. To do this, we add a `+` after the `\d` expression to modify the meaning to one or more.
2. Change the regular expression, as shown highlighted in the following code:

```
Regex ageChecker = new(@"^\d+$");
```

1. Run the code again and note the regular expression only allows zero or positive whole numbers of any length.

Regular expression performance improvements

The .NET types for working with regular expressions are used throughout the .NET platform and many of the apps built with it. As such, they have a significant impact on performance, but until now, they have not received much optimization attention from Microsoft.

With .NET 5 and later, the `System.Text.RegularExpressions` namespace has rewritten internals to squeeze out maximum performance. Common regular expression benchmarks using methods like `IsMatch` are now five times faster. And the best thing is, you do not have to change your code to get the benefits!

Understanding the syntax of a regular expression

Here are some common regular expression symbols that you can use in regular expressions:

Symbol	Meaning	Symbol	Meaning
`^`	Start of input	`$`	End of input
`\d`	A single digit	`\D`	A single NON-digit
`\s`	Whitespace	`\S`	NON-whitespace
`\w`	Word characters	`\W`	NON-word characters
`[A-Za-z0-9]`	Range(s) of characters	`\^`	^ (caret) character
`[aeiou]`	Set of characters	`[^aeiou]`	NOT in a set of characters
`.`	Any single character	`\.`	. (dot) character

In addition, here are some regular expression quantifiers that affect the previous symbols in a regular expression:

Symbol	Meaning	Symbol	Meaning
`+`	One or more	`?`	One or none
`{3}`	Exactly three	`{3,5}`	Three to five
`{3,}`	At least three	`{,3}`	Up to three

Examples of regular expressions

Here are some examples of regular expressions with a description of their meaning:

Expression	Meaning
`\d`	A single digit somewhere in the input.
`a`	The character a somewhere in the input.
`Bob`	The word Bob somewhere in the input.
`^Bob`	The word Bob at the start of the input.
`Bob$`	The word Bob at the end of the input.
`^\d{2}$`	Exactly two digits.
`^[0-9]{2}$`	Exactly two digits.
`^[A-Z]{4,}$`	At least four uppercase English letters in the ASCII character set only.
`^[A-Za-z]{4,}$`	At least four upper or lowercase English letters in the ASCII character set only.

`^[A-Z]{2}\d{3}$`	Two uppercase English letters in the ASCII character set and three digits only.
`^[A-Za-z\u00c0-\u017e]+$`	At least one uppercase or lowercase English letter in the ASCII character set or European letters in the Unicode character set, as shown in the following list: `ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝ` `Þßàáâãäåæçèéêëìíîïðñòóôõö÷øùúûüýþÿıŒœŠšŸŽž`
`^d.g$`	The letter `d` , then any character, and then the letter `g` , so it would match both `dig` and `dog` or any single character between the `d` and `g` .
`^d\.g$`	The letter `d` , then a dot `.` , and then the letter `g` , so it would match `d.g` only.

Good Practice: Use regular expressions to validate input from the user. The same regular expressions can be reused in other languages such as JavaScript and Python.

Splitting a complex comma-separated string

Earlier in this chapter, you learned how to split a simple comma-separated string variable. But what about the following example of film titles?

```
"Monsters, Inc.","I, Tonya","Lock, Stock and Two Smoking Barrels"
```

The `string` value uses double quotes around each film title. We can use these to identify whether we need to split on a comma (or not). The `Split` method is not powerful enough, so we can use a regular expression instead.

Good Practice: You can read a fuller explanation in the Stack Overflow article that inspired this task at the following link: https://stackoverflow.com/questions/18144431/regex-to-split-a-csv

To include double quotes inside a `string` value, we prefix them with a backslash, or we could use the C# 11 raw string literal feature in C# 11 or later:

1. Add statements to store a complex comma-separated `string` variable, and then split it in a dumb way using the `Split` method, as shown in the following code:

```
// C# 1 to 10: Use escaped double-quote characters \"
// string films = "\"Monsters, Inc.\",\"I, Tonya\",\"Lock, Stock and Two Smoking Barrels\"";
// C# 11 or later: Use """ to start and end a raw string literal
string films = """
"Monsters, Inc.","I, Tonya","Lock, Stock and Two Smoking Barrels"
""";
WriteLine($"Films to split: {films}");
string[] filmsDumb = films.Split(',');
WriteLine("Splitting with string.Split method:");
foreach (string film in filmsDumb)
{
  WriteLine(film);
}
```

1. Add statements to define a regular expression to split and write the film titles in a smart way, as shown in the following code:

```
Regex csv = new(
  "(?:^|,)(?=[^\"]|(\")?)\"?((?(1)[^\"]*|[^,\"]*))\"?(?=,|$)");
MatchCollection filmsSmart = csv.Matches(films);
WriteLine("Splitting with regular expression:");
foreach (Match film in filmsSmart)
{
  WriteLine(film.Groups[2].Value);
}
```

1. Run the code and view the result, as shown in the following output:

```
Splitting with string.Split method:
"Monsters
 Inc."
"I
 Tonya"
"Lock
 Stock and Two Smoking Barrels"
Splitting with regular expression:
Monsters, Inc.
I, Tonya
Lock, Stock and Two Smoking Barrels
```

Storing multiple objects in collections

Another of the most common types of data is collections. If you need to store multiple values in a variable, then you can use a collection.

A collection is a data structure in memory that can manage multiple items in different ways, although all collections have some shared functionality.

The most common types in .NET for working with collections are shown in the following table:

Namespace	Example type(s)	Description
`System.Collections`	`IEnumerable`, `IEnumerable<T>`	Interfaces and base classes used by collections.
`System.Collections.Generic`	`List<T>`, `Dictionary<T>`, `Queue<T>`, `Stack<T>`	Introduced in C# 2.0 with .NET Framework 2.0. These collections allow you to specify the type you want to store using a generic type parameter (which is safer, faster, and more efficient).
`System.Collections.Concurrent`	`BlockingCollection`, `ConcurrentDictionary`, `ConcurrentQueue`	These collections are safe to use in multithreaded scenarios.
`System.Collections.Immutable`	`ImmutableArray`, `ImmutableDictionary`, `ImmutableList`, `ImmutableQueue`	Designed for scenarios where the contents of the original collection will never change, although they can create modified collections as a new instance.

Common features of all collections

All collections implement the `ICollection` interface; this means that they must have a `Count` property to tell you how many objects are in them, as shown in the following code:

```
namespace System.Collections;
public interface ICollection : IEnumerable
{
  int Count { get; }
  bool IsSynchronized { get; }
  object SyncRoot { get; }
  void CopyTo(Array array, int index);
}
```

For example, if we had a collection named `passengers`, we could do this:

```
int howMany = passengers.Count;
```

All collections implement the `IEnumerable` interface, which means that they can be iterated using the `foreach` statement. They must have a `GetEnumerator` method that returns an object that implements `IEnumerator`; this means that the returned `object` must have `MoveNext` and `Reset` methods for navigating through the collection and a `Current` property containing the current item in the collection, as shown in the following code:

```
namespace System.Collections;
public interface IEnumerable
{
  IEnumerator GetEnumerator();
}
public interface IEnumerator
{
  object Current { get; }
  bool MoveNext();
  void Reset();
}
```

For example, to perform an action on each object in the `passengers` collection, we could write the following code:

```
foreach (Passenger p in passengers)
{
  // perform an action on each passenger
}
```

As well as `object`-based collection interfaces, there are also generic interfaces and classes, where the generic type defines the type stored in the collection, as shown in the following code:

```
namespace System.Collections.Generic;
public interface ICollection<T> : IEnumerable<T>, IEnumerable
{
  int Count { get; }
  bool IsReadOnly { get; }
  void Add(T item);
  void Clear();
  bool Contains(T item);
  void CopyTo(T[] array, int index);
  bool Remove(T item);
}
```

Improving performance by ensuring the capacity of a collection

Since .NET 1.1, types like `StringBuilder` have had a method named `EnsureCapacity` that can presize its internal storage array to the expected final size of the `string`. This improves performance because it does not have to repeatedly increment the size of the array as more characters are appended.

Since .NET Core 2.1, types like `Dictionary<T>` and `HashSet<T>` have also had `EnsureCapacity`.

In .NET 6 and later, collections like `List<T>`, `Queue<T>`, and `Stack<T>` now have an `EnsureCapacity` method too, as shown in the following code:

```
List<string> names = new();
names.EnsureCapacity(10_000);
// load ten thousand names into the list
```

Understanding collection choices

There are several different choices of collection that you can use for different purposes: lists, dictionaries, stacks, queues, sets, and many other more specialized collections.

Lists

Lists, that is, a type that implements `IList<T>`, are **ordered collections**, as shown in the following code:

```
namespace System.Collections.Generic;
[DefaultMember("Item")] // aka this indexer
public interface IList<T> : ICollection<T>, IEnumerable<T>, IEnumerable
{
  T this[int index] { get; set; }
  int IndexOf(T item);
  void Insert(int index, T item);
  void RemoveAt(int index);
}
```

`IList<T>` derives from `ICollection<T>` so it has a `Count` property, and an `Add` method to put an item at the end of the collection, as well as an `Insert` method to put an item in the list at a specified position, and `RemoveAt` to remove an item at a specified position.

Lists are a good choice when you want to manually control the order of items in a collection. Each item in a list has a unique index (or position) that is automatically assigned. Items can be any type defined by `T` and items can be duplicated. Indexes are `int` types and start from `0`, so the first item in a list is at index `0`, as shown in the following table:

Index	Item
0	London
1	Paris
2	London
3	Sydney

If a new item (for example, Santiago) is inserted between London and Sydney, then the index of Sydney is automatically incremented. Therefore, you must be aware that an item's index can change after inserting or removing items, as shown in the following table:

Index	Item
0	London
1	Paris
2	London
3	Santiago
4	Sydney

Dictionaries

Dictionaries are a good choice when each **value** (or object) has a unique sub value (or a made-up value) that can be used as a **key** to quickly find a value in the collection later. The key must be unique. For example, if you are storing a list of people, you could choose to use a government-issued identity number as the key. Dictionaries are called **hashmaps** in other languages like Python and Java.

Think of the key as being like an index entry in a real-world dictionary. It allows you to quickly find the definition of a word because the words (in other words, keys) are kept sorted, and if we know we're looking for the definition of *manatee*, we would jump to the middle of the dictionary to start looking, because the letter *M* is in the middle of the alphabet.

Dictionaries in programming are similarly smart when looking something up. They must implement the interface `IDictionary<TKey, TValue>`, as shown in the following code:

```
namespace System.Collections.Generic;
[DefaultMember("Item")] // aka this indexer
public interface IDictionary<TKey, TValue>
  : ICollection<KeyValuePair<TKey, TValue>>,
    IEnumerable<KeyValuePair<TKey, TValue>>, IEnumerable
{
  TValue this[TKey key] { get; set; }
  ICollection<TKey> Keys { get; }
  ICollection<TValue> Values { get; }
  void Add(TKey key, TValue value);
  bool ContainsKey(TKey key);
  bool Remove(TKey key);
  bool TryGetValue(TKey key, [MaybeNullWhen(false)] out TValue value);
}
```

Items in a dictionary are instances of the `struct`, aka the value type, `KeyValuePair<TKey, TValue>`, where `TKey` is the type of the key and `TValue` is the type of the value, as shown in the following code:

```
namespace System.Collections.Generic;
public readonly struct KeyValuePair<TKey, TValue>
{
  public KeyValuePair(TKey key, TValue value);
  public TKey Key { get; }
  public TValue Value { get; }
  [EditorBrowsable(EditorBrowsableState.Never)]
  public void Deconstruct(out TKey key, out TValue value);
  public override string ToString();
}
```

An example `Dictionary<string, Person>` uses a `string` as the key and a `Person` instance as the value. `Dictionary<string, string>` uses `string` values for both, as shown in the following table:

Key	Value
BSA	Bob Smith
MW	Max Williams
BSB	Bob Smith
AM	Amir Mohammed

Stacks

Stacks are a good choice when you want to implement **last-in, first-out** (**LIFO**) behavior. With a stack, you can only directly access or remove the one item at the top of the stack, although you can enumerate to read through the whole stack of items. You cannot, for example, directly access the second item in a stack.

For example, word processors use a stack to remember the sequence of actions you have recently performed, and then when you press Ctrl + Z, it will undo the last action in the stack, and then the next-to-last action, and so on.

Queues

Queues are a good choice when you want to implement the **first-in, first-out** (**FIFO**) behavior. With a queue, you can only directly access or remove the one item at the front of the queue, although you can enumerate to read through the whole queue of items. You cannot, for example, directly access the second item in a queue.

For example, background processes use a queue to process work items in the order that they arrive, just like people standing in line at the post office.

.NET 6 introduced the `PriorityQueue`, where each item in the queue has a priority value assigned as well as their position in the queue.

Sets

Sets are a good choice when you want to perform set operations between two collections. For example, you may have two collections of city names, and you want to know which names appear in both sets (known as the *intersect* between the sets). Items in a set must be unique.

Collection methods summary

Each collection has a different set of methods for adding and removing items, as shown in the following table:

Collection	Add methods	Remove methods	Description
`List`	`Add`, `Insert`	`Remove`, `RemoveAt`	Lists are ordered so items have an integer index position. `Add` will add a new item at the end of the list. `Insert` will add a new item at the index position specified.
`Dictionary`	`Add`	`Remove`	Dictionaries are not ordered so items do not have integer index positions. You can check if a key has been used by calling the `ContainsKey` method.
`Stack`	`Push`	`Pop`	Stacks always add a new item at the top of the stack using the `Push` method. The first item is at the bottom. Items are always removed from the top of the stack using the `Pop` method. Call the `Peek` method to see this value without removing it.
`Queue`	`Enqueue`	`Dequeue`	Queues always add a new item at the end of the queue using the `Enqueue` method. The first item is at the front of the queue. Items are always removed from the front of the queue using the `Dequeue` method. Call the `Peek` method to see this value without removing it.

Working with lists

Let's explore lists:

1. Use your preferred code editor to add a new **Console App** / `console` project named `WorkingWithCollections` to the `Chapter08` solution/workspace.
    - In Visual Studio Code, select `WorkingWithCollections` as the active OmniSharp project.
2. Add a new class file named `Program.Helpers.cs`.

3. In `Program.Helpers.cs`, define a partial `Program` class with a method to output a collection of `string` values with a title, as shown in the following code:

```
partial class Program
{
  static void Output(string title, IEnumerable<string> collection)
  {
    WriteLine(title);
    foreach (string item in collection)
    {
      WriteLine($"  {item}");
    }
  }
}
```

1. In `Program.cs`, delete the existing statements and then add some statements to illustrate some of the common ways of defining and working with lists, as shown in the following code:

```
// Simple syntax for creating a list and adding three items
List<string> cities = new();
cities.Add("London");
cities.Add("Paris");
cities.Add("Milan");
/* Alternative syntax that is converted by the compiler into
   the three Add method calls above
List<string> cities = new()
  { "London", "Paris", "Milan" }; */
/* Alternative syntax that passes an
   array of string values to AddRange method
List<string> cities = new();
cities.AddRange(new[] { "London", "Paris", "Milan" }); */
Output("Initial list", cities);
WriteLine($"The first city is {cities[0]}.");
WriteLine($"The last city is {cities[cities.Count - 1]}.");
cities.Insert(0, "Sydney");
Output("After inserting Sydney at index 0", cities);
cities.RemoveAt(1);
cities.Remove("Milan");
Output("After removing two cities", cities);
```

1. Run the code and view the result, as shown in the following output:

```
Initial list
  London
  Paris
  Milan
The first city is London.
The last city is Milan.
After inserting Sydney at index 0
  Sydney
  London
  Paris
  Milan
After removing two cities
  Sydney
  Paris
```

Working with dictionaries

Let's explore dictionaries:

1. In `Program.cs`, add some statements to illustrate some of the common ways of working with dictionaries, for example, looking up word definitions, as shown in the following code:

```
Dictionary<string, string> keywords = new();
// add using named parameters
keywords.Add(key: "int", value: "32-bit integer data type");
// add using positional parameters
keywords.Add("long", "64-bit integer data type");
keywords.Add("float", "Single precision floating point number");
/* Alternative syntax; compiler converts this to calls to Add method
Dictionary<string, string> keywords = new()
{
  { "int", "32-bit integer data type" },
  { "long", "64-bit integer data type" },
  { "float", "Single precision floating point number" },
}; */
/* Alternative syntax; compiler converts this to calls to Add method
Dictionary<string, string> keywords = new()
{
  ["int"] = "32-bit integer data type",
  ["long"] = "64-bit integer data type",
  ["float"] = "Single precision floating point number", // last comma is optional
}; */
Output("Dictionary keys:", keywords.Keys);
Output("Dictionary values:", keywords.Values);
WriteLine("Keywords and their definitions");
foreach (KeyValuePair<string, string> item in keywords)
{
  WriteLine($"  {item.Key}: {item.Value}");
}
// look up a value using a key
string key = "long";
WriteLine($"The definition of {key} is {keywords[key]}");
```

1. Run the code and view the result, as shown in the following output:

```
Dictionary keys:
  int
  long
  float
Dictionary values:
  32-bit integer data type
  64-bit integer data type
  Single precision floating point number
Keywords and their definitions
  int: 32-bit integer data type
  long: 64-bit integer data type
  float: Single precision floating point number
The definition of long is 64-bit integer data type
```

Working with queues

Let's explore queues:

1. In `Program.cs`, add some statements to illustrate some of the common ways of working with queues, for example, handling customers in a queue for coffee, as shown in the following code:

```
Queue<string> coffee = new();
coffee.Enqueue("Damir"); // front of queue
coffee.Enqueue("Andrea");
coffee.Enqueue("Ronald");
coffee.Enqueue("Amin");
coffee.Enqueue("Irina"); // back of queue
Output("Initial queue from front to back", coffee);
// server handles next person in queue
string served = coffee.Dequeue();
WriteLine($"Served: {served}.");
// server handles next person in queue
served = coffee.Dequeue();
```

```
WriteLine($"Served: {served}.");
Output("Current queue from front to back", coffee);
WriteLine($"{coffee.Peek()} is next in line.");
Output("Current queue from front to back", coffee);
```

1. Run the code and view the result, as shown in the following output:

```
Initial queue from front to back
  Damir
  Andrea
  Ronald
  Amin
  Irina
Served: Damir.
Served: Andrea.
Current queue from front to back
  Ronald
  Amin
  Irina
Ronald is next in line.
Current queue from front to back
  Ronald
  Amin
  Irina
```

1. In `Program.Helpers.cs`, in the partial `Program` class, add a static method named `OutputPQ`, as shown in the following code:

```
static void OutputPQ<TElement, TPriority>(string title,
  IEnumerable<(TElement Element, TPriority Priority)> collection)
{
  WriteLine(title);
  foreach ((TElement, TPriority) item in collection)
  {
    WriteLine($"  {item.Item1}: {item.Item2}");
  }
}
```

Note that the `OutputPQ` method is generic. You can specify the two types used in the tuples that are passed in as `collection`.

1. In `Program.cs`, add some statements to illustrate some of the common ways of working with priority queues, as shown in the following code:

```
PriorityQueue<string, int> vaccine = new();
// add some people
// 1 = high priority people in their 70s or poor health
// 2 = medium priority e.g. middle-aged
// 3 = low priority e.g. teens and twenties
vaccine.Enqueue("Pamela", 1);  // my mum (70s)
vaccine.Enqueue("Rebecca", 3); // my niece (teens)
vaccine.Enqueue("Juliet", 2);  // my sister (40s)
vaccine.Enqueue("Ian", 1);     // my dad (70s)
OutputPQ("Current queue for vaccination:", vaccine.UnorderedItems);
WriteLine($"{vaccine.Dequeue()} has been vaccinated.");
WriteLine($"{vaccine.Dequeue()} has been vaccinated.");
OutputPQ("Current queue for vaccination:", vaccine.UnorderedItems);
WriteLine($"{vaccine.Dequeue()} has been vaccinated.");
WriteLine("Adding Mark to queue with priority 2");
vaccine.Enqueue("Mark", 2); // me (40s)
WriteLine($"{vaccine.Peek()} will be next to be vaccinated.");
OutputPQ("Current queue for vaccination:", vaccine.UnorderedItems);
```

1. Run the code and view the result, as shown in the following output:

```
Current queue for vaccination:
  Pamela: 1
  Rebecca: 3
  Juliet: 2
  Ian: 1
Pamela has been vaccinated.
Ian has been vaccinated.
Current queue for vaccination:
  Juliet: 2
  Rebecca: 3
Juliet has been vaccinated.
Adding Mark to queue with priority 2
Mark will be next to be vaccinated.
Current queue for vaccination:
  Mark: 2
  Rebecca: 3
```

Sorting collections

A `List<T>` class can be sorted by manually calling its `Sort` method (but remember that the indexes of each item will change). Manually sorting a list of `string` values or other built-in types will work without extra effort on your part, but if you create a collection of your own type, then that type must implement an interface named `IComparable`. You learned how to do this in *Chapter 6*, *Implementing Interfaces and Inheriting Classes*.

A `Stack<T>` or `Queue<T>` collection cannot be sorted because you wouldn't usually want that functionality; for example, you would probably never sort a queue of guests checking into a hotel. But sometimes, you might want to sort a dictionary or a set.

Sometimes it would be useful to have an automatically sorted collection, that is, one that maintains the items in a sorted order as you add and remove them.

There are multiple auto-sorting collections to choose from. The differences between these sorted collections are often subtle but can have an impact on the memory requirements and performance of your application, so it is worth putting effort into picking the most appropriate option for your requirements.

Some common auto-sorting collections are shown in the following table:

Collection	Description
`SortedDictionary<TKey, TValue>`	This represents a collection of key/value pairs that are sorted by key.
`SortedList<TKey, TValue>`	This represents a collection of key/value pairs that are sorted by key.
`SortedSet<T>`	This represents a collection of unique objects that are maintained in a sorted order.

More specialized collections

There are a few other collections for special situations.

Working with a compact array of bit values

The `System.Collections.BitArray` collection manages a compact array of bit values, which are represented as Booleans, where `true` indicates that the bit is on (value is 1) and `false` indicates the bit is off (value is 0).

Working with efficient lists

The `System.Collections.Generics.LinkedList<T>` collection represents a doubly linked list where every item has a reference to its previous and next items. They provide better performance compared to `List<T>` for scenarios where you will frequently insert and remove items from the middle of the list. In a `LinkedList<T>` the items do not have to be rearranged in memory.

Working with immutable collections

Sometimes you need to make a collection immutable, meaning that its members cannot change; that is, you cannot add or remove them.

If you import the `System.Collections.Immutable` namespace, then any collection that implements `IEnumerable<T>` is given six extension methods to convert it into an immutable list, dictionary, hash set, and so on.

Let's see a simple example:

1. In the `WorkingWithCollections` project, in `Program.cs`, import the `System.Collections.Immutable` namespace.
2. In `Program.cs`, add statements to convert the cities list into an immutable list and then add a new city to it, as shown in the following code:

```
ImmutableList<string> immutableCities = cities.ToImmutableList();
ImmutableList<string> newList = immutableCities.Add("Rio");
Output("Immutable list of cities:", immutableCities);
Output("New list of cities:", newList);
```

1. Run the code, view the result, and note that the immutable list of cities does not get modified when you call the `Add` method on it; instead, it returns a new list with all the existing cities plus the newly added city, as shown in the following output:

```
Immutable list of cities:
  Sydney
  Paris
New list of cities:
  Sydney
  Paris
  Rio
```

Good Practice: To improve performance, many applications store a shared copy of commonly accessed objects in a central cache. To safely allow multiple threads to work with those objects knowing they won't change, you should make them immutable or use a concurrent collection type that you can read about at the following link: https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent

Good practice with collections

Let's say you need to create a method to process a collection. For maximum flexibility, you could declare the input parameter to be `IEnumerable<T>` and make the method generic, as shown in the following code:

```
void ProcessCollection<T>(IEnumerable<T> collection)
{
  // process the items in the collection,
  // perhaps using a foreach statement
}
```

I could pass an array, a list, a queue, a stack, containing any type like `int` or `string` or `Person`, or anything else that implements `IEnumerable<T>` into this method and it will process the items. However, the flexibility to pass any collection to this method comes at a performance cost.

One of the performance problems with `IEnumerable<T>` is also one of its benefits: deferred execution, also known as lazy loading. Types that implement this interface do not have to implement deferred execution, but many do.

But the worst performance problem with `IEnumerable<T>` is that the iteration must allocate an object on the heap. To avoid this memory allocation, you should define your method using a concrete type, as shown highlighted in the following code:

```
void ProcessCollection<T>(List<T> collection)
{
  // process the items in the collection,
  // perhaps using a foreach statement
}
```

This will use the `List<T>.Enumerator GetEnumerator()` method that returns a `struct` instead of the `IEnumerator<T> GetEnumerator()` method that returns a reference type. Your code will be two to three times faster and require less memory. As with all recommendations related to performance, you should confirm the benefit by running performance tests on your actual code in a product environment.

Working with spans, indexes, and ranges

One of Microsoft's goals with .NET Core 2.1 was to improve performance and resource usage. A key .NET feature that enables this is the `Span<T>` type.

Using memory efficiently using spans

When manipulating arrays, you will often create new copies of subsets of existing ones so that you can process just the subset. This is not efficient because duplicate objects must be created in memory.

If you need to work with a subset of an array, use a **span** because it is like a window into the original array. This is more efficient in terms of memory usage and improves performance. Spans only work with arrays, not collections, because the memory must be contiguous.

Before we look at spans in more detail, we need to understand some related objects: indexes and ranges.

Identifying positions with the Index type

C# 8.0 introduced two features for identifying an item's index position within an array and a range of items using two indexes.

You learned in the previous topic that objects in a list can be accessed by passing an integer into their indexer, as shown in the following code:

```
int index = 3;
Person p = people[index]; // fourth person in array
char letter = name[index]; // fourth letter in name
```

The `Index` value type is a more formal way of identifying a position, and supports counting from the end, as shown in the following code:

```
// two ways to define the same index, 3 in from the start
Index i1 = new(value: 3); // counts from the start
Index i2 = 3; // using implicit int conversion operator
// two ways to define the same index, 5 in from the end
Index i3 = new(value: 5, fromEnd: true);
Index i4 = ^5; // using the caret operator
```

Identifying ranges with the Range type

The `Range` value type uses `Index` values to indicate the start and end of its range, using its constructor, C# syntax, or its static methods, as shown in the following code:

```
Range r1 = new(start: new Index(3), end: new Index(7));
Range r2 = new(start: 3, end: 7); // using implicit int conversion
Range r3 = 3..7; // using C# 8.0 or later syntax
Range r4 = Range.StartAt(3); // from index 3 to last index
Range r5 = 3..; // from index 3 to last index
Range r6 = Range.EndAt(3); // from index 0 to index 3
Range r7 = ..3; // from index 0 to index 3
```

Extension methods have been added to `string` values (that internally use an array of `char`), `int` arrays, and spans to make ranges easier to work with. These extension methods accept a range as a parameter and return a `Span<T>`. This makes them very memory-efficient.

Using indexes, ranges, and spans

Let's explore using indexes and ranges to return spans:

1. Use your preferred code editor to add a new **Console App** / `console` project named `WorkingWithRanges` to the `Chapter08` solution/workspace.
   - In Visual Studio Code, select `WorkingWithRanges` as the active OmniSharp project.
2. In `Program.cs`, delete the existing statements and then add statements to compare using the `string` type's `Substring` method with ranges to extract parts of someone's name, as shown in the following code:

```
string name = "Samantha Jones";
// getting the lengths of the first and last names
int lengthOfFirst = name.IndexOf(' ');
int lengthOfLast = name.Length - lengthOfFirst - 1;
// Using Substring
string firstName = name.Substring(
  startIndex: 0,
  length: lengthOfFirst);
string lastName = name.Substring(
  startIndex: name.Length - lengthOfLast,
  length: lengthOfLast);
WriteLine($"First name: {firstName}, Last name: {lastName}");
// Using spans
ReadOnlySpan<char> nameAsSpan = name.AsSpan();
ReadOnlySpan<char> firstNameSpan = nameAsSpan[0..lengthOfFirst];
ReadOnlySpan<char> lastNameSpan = nameAsSpan[^lengthOfLast..^0];
WriteLine("First name: {0}, Last name: {1}",
  arg0: firstNameSpan.ToString(),
  arg1: lastNameSpan.ToString());
```

1. Run the code and view the result, as shown in the following output:

```
First name: Samantha, Last name: Jones
First name: Samantha, Last name: Jones
```

Working with network resources

Sometimes you will need to work with network resources. The most common types in .NET for working with network resources are shown in the following table:

Namespace	Example type(s)	Description

`System.Net`	`Dns` , `Uri` , `Cookie` , `WebClient` , `IPAddress`	These are for working with DNS servers, URIs, IP addresses, and so on.
`System.Net`	`FtpStatusCode` , `FtpWebRequest` , `FtpWebResponse`	These are for working with FTP servers.
`System.Net`	`HttpStatusCode` , `HttpWebRequest` , `HttpWebResponse`	These are for working with HTTP servers; that is, websites and services. Types from `System.Net.Http` are easier to use.
`System.Net.Http`	`HttpClient` , `HttpMethod` , `HttpRequestMessage` , `HttpResponseMessage`	These are for working with HTTP servers; that is, websites and services. You will learn how to use these in *Chapter 15* , *Building and Consuming Web Services* .
`System.Net.Mail`	`Attachment` , `MailAddress` , `MailMessage` , `SmtpClient`	These are for working with SMTP servers; that is, sending email messages.
`System.Net .NetworkInformation`	`IPStatus` , `NetworkChange` , `Ping` , `TcpStatistics`	These are for working with low-level network protocols.

Working with URIs, DNS, and IP addresses

Let's explore some common types for working with network resources:

1. Use your preferred code editor to add a new **Console App** / `console` project named `WorkingWithNetworkResources` to the `Chapter08` solution/workspace.
   - In Visual Studio Code, select `WorkingWithNetworkResources` as the active OmniSharp project.
2. In `Program.cs` , delete the existing statements and then import the namespace for working with a network, as shown in the following code:

```
using System.Net; // IPHostEntry, Dns, IPAddress
```

1. In `Program.cs` , add statements to prompt the user to enter a website address, and then use the `Uri` type to break it down into its parts, including the scheme (HTTP, FTP, and so on), port number, and host, as shown in the following code:

```
Write("Enter a valid web address (or press Enter): ");
string? url = ReadLine();
if (string.IsNullOrWhiteSpace(url)) // if they enter nothing...
{
  // ... set a default URL
  url = "https://stackoverflow.com/search?q=securestring";
}
Uri uri = new(url);
WriteLine($"URL: {url}");
WriteLine($"Scheme: {uri.Scheme}");
WriteLine($"Port: {uri.Port}");
WriteLine($"Host: {uri.Host}");
```

```
WriteLine($"Path: {uri.AbsolutePath}");
WriteLine($"Query: {uri.Query}");
```

1. Run the code, enter a valid website address or press Enter, and view the result, as shown in the following output:

```
Enter a valid web address (or press Enter):
URL: https://stackoverflow.com/search?q=securestring
Scheme: https
Port: 443
Host: stackoverflow.com
Path: /search
Query: ?q=securestring
```

1. In `Program.cs`, add statements to get the IP address for the entered website, as shown in the following code:

```
IPHostEntry entry = Dns.GetHostEntry(uri.Host);
WriteLine($"{entry.HostName} has the following IP addresses:");
foreach (IPAddress address in entry.AddressList)
{
  WriteLine($"  {address} ({address.AddressFamily})");
}
```

1. Run the code, enter a valid website address or press Enter, and view the result, as shown in the following output:

```
stackoverflow.com has the following IP addresses:
  151.101.1.69 (InterNetwork)
  151.101.65.69 (InterNetwork)
  151.101.129.69 (InterNetwork)
  151.101.193.69 (InterNetwork)
```

Pinging a server

Now you will add code to ping a web server to check its health:

1. In `Program.cs`, import the namespace to get more information about networks, as shown in the following code:

```
using System.Net.NetworkInformation; // Ping, PingReply, IPStatus
```

1. Add statements to ping the entered website, as shown in the following code:

```
try
{
  Ping ping = new();
  WriteLine("Pinging server. Please wait...");
  PingReply reply = ping.Send(uri.Host);
  WriteLine($"{uri.Host} was pinged and replied: {reply.Status}.");
  if (reply.Status == IPStatus.Success)
  {
    WriteLine("Reply from {0} took {1:N0}ms",
      arg0: reply.Address,
      arg1: reply.RoundtripTime);
  }
}
catch (Exception ex)
{
  WriteLine($"{ex.GetType().ToString()} says {ex.Message}");
}
```

1. Run the code, press Enter, and view the result, as shown in the following output:

```
Pinging server. Please wait...
stackoverflow.com was pinged and replied: Success.
Reply from 151.101.193.69 took 9ms
```

1. Run the code again but this time enter http://google.com, as shown in the following output:

```
Enter a valid web address (or press Enter): http://google.com
URL: http://google.com
Scheme: http
Port: 80
Host: google.com
Path: /
Query:
google.com has the following IP addresses:
  2a00:1450:4009:822::200e (InterNetworkV6)
  142.250.180.14 (InterNetwork)
Pinging server. Please wait...
google.com was pinged and replied: Success.
Reply from 2a00:1450:4009:822::200e took 9ms
```

Practicing and exploring

Test your knowledge and understanding by answering some questions, getting some hands-on practice, and exploring with deeper research into the topics in this chapter.

Exercise 8.1 – Test your knowledge

Use the web to answer the following questions:

1. What is the maximum number of characters that can be stored in a `string` variable?
2. When and why should you use a `SecureString` type?
3. When is it appropriate to use a `StringBuilder` class?
4. When should you use a `LinkedList<T>` class?
5. When should you use a `SortedDictionary<T>` class rather than a `SortedList<T>` class?
6. In a regular expression, what does `$` mean?
7. In a regular expression, how can you represent digits?
8. Why should you *not* use the official standard for email addresses to create a regular expression to validate a user's email address?
9. What characters are output when the following code runs?

```
string city = "Aberdeen";
ReadOnlySpan<char> citySpan = city.AsSpan()[^5..^0];
WriteLine(citySpan.ToString());
```

1. How could you check that a web service is available before calling it?

Exercise 8.2 – Practice regular expressions

In the `Chapter08` solution/workspace, create a console app named `Ch08Ex02RegularExpressions` that prompts the user to enter a regular expression and then prompts the user to enter some input and compare the two for a match until the user presses *Esc*, as shown in the following output:

```
The default regular expression checks for at least one digit.
Enter a regular expression (or press ENTER to use the default): ^[a-z]+$
Enter some input: apples
apples matches ^[a-z]+$? True
Press ESC to end or any key to try again.
Enter a regular expression (or press ENTER to use the default): ^[a-z]+$
Enter some input: abc123xyz
```

```
abc123xyz matches ^[a-z]+$? False
Press ESC to end or any key to try again.
```

Exercise 8.3 – Practice writing extension methods

In the `Chapter08` solution/workspace, create a class library named `Ch08Ex03NumbersAsWordsLib` and projects to test it. It should define extension methods that extend number types such as `BigInteger` and `int` with a method named `ToWords` that returns a `string` describing the number.

For example, `18,000,000` would be eighteen million, and `18,456,002,032,011,000,007` would be eighteen quintillion, four hundred and fifty-six quadrillion, two trillion, thirty-two billion, eleven million, and seven.

You can read more about names for large numbers at the following link:
https://en.wikipedia.org/wiki/Names_of_large_numbers

Exercise 8.4 – Explore topics

Use the links on the following page to learn more detail about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-8---working-with-common-net-types

Summary

In this chapter, you explored:

- Choices for types to store and manipulate numbers.
- Handling text, including using regular expressions for validating input.
- Collections to use for storing multiple items.
- Worked with indexes, ranges, and spans.
- Used some types for interacting with network resources.

In the next chapter, we will manage files and streams, encode and decode text, and perform serialization.

9 Working with Files, Streams, and Serialization

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about reading and writing to files and streams, text encoding, and serialization.

We will cover the following topics:

- Managing the filesystem
- Reading and writing with streams
- Encoding and decoding text
- Reading and writing with random access handles
- Serializing object graphs

Managing the filesystem

Your applications will often need to perform input and output operations with files and directories in different environments. The `System` and `System.IO` namespaces contain classes for this purpose.

Handling cross-platform environments and filesystems

Let's explore how to handle cross-platform environments and the differences between Windows and Linux or macOS. Paths are different for Windows, macOS, and Linux, so we will start by exploring how .NET handles this:

1. Use your preferred code editor to create a new project, as defined in the following list:
   - Project template: **Console App** / `console`
   - Project file and folder: `WorkingWithFileSystems`
   - Workspace/solution file and folder: `Chapter09`
2. In the project file, add an element to statically and globally import the `System.Console` class.
3. Add a new class file named `Program.Helpers.cs`.
4. In `Program.Helpers.cs`, add a partial `Program` class with a `SectionTitle` method, as shown in the following code:

```
partial class Program
{
  static void SectionTitle(string title)
  {
    ConsoleColor previousColor = ForegroundColor;
    ForegroundColor = ConsoleColor.Yellow;
    WriteLine("*");
    WriteLine($"* {title}");
    WriteLine("*");
    ForegroundColor = previousColor;
  }
}
```

1. In `Program.cs`, delete the existing statements and then add statements to statically import the `System.IO.Directory`, `System.Environment`, and `System.IO.Path` types, as shown in the following code:

```
using static System.IO.Directory;
using static System.IO.Path;
using static System.Environment;
```

1. In `Program.cs`, add statements to do the following:
   - Output the path and directory separation characters.
   - Output the path of the current directory.
   - Output some special paths for system files, temporary files, and documents.

```
SectionTitle("* Handling cross-platform environments and filesystems");
WriteLine("{0,-33} {1}", arg0: "Path.PathSeparator",
  arg1: PathSeparator);
WriteLine("{0,-33} {1}", arg0: "Path.DirectorySeparatorChar",
  arg1: DirectorySeparatorChar);
WriteLine("{0,-33} {1}", arg0: "Directory.GetCurrentDirectory()",
  arg1: GetCurrentDirectory());
WriteLine("{0,-33} {1}", arg0: "Environment.CurrentDirectory",
  arg1: CurrentDirectory);
WriteLine("{0,-33} {1}", arg0: "Environment.SystemDirectory",
  arg1: SystemDirectory);
WriteLine("{0,-33} {1}", arg0: "Path.GetTempPath()",
  arg1: GetTempPath());
WriteLine("GetFolderPath(SpecialFolder");
WriteLine("{0,-33} {1}", arg0: " .System)",
  arg1: GetFolderPath(SpecialFolder.System));
WriteLine("{0,-33} {1}", arg0: " .ApplicationData)",
  arg1: GetFolderPath(SpecialFolder.ApplicationData));
WriteLine("{0,-33} {1}", arg0: " .MyDocuments)",
  arg1: GetFolderPath(SpecialFolder.MyDocuments));
WriteLine("{0,-33} {1}", arg0: " .Personal)",
  arg1: GetFolderPath(SpecialFolder.Personal));
```

The `Environment` type has many other useful members that we did not use in this code, including the `GetEnvironmentVariables` method and the `OSVersion` and `ProcessorCount` properties.

1. Run the code and view the result, as shown in *Figure 9.1*:

Figure 9.1: Running your application to show filesystem information on Windows with Visual Studio 2022

When running the console app using `dotnet run` with Visual Studio Code, the `CurrentDirectory` will be the project folder, not a folder inside `bin`, as shown in the following output:

```
Path.PathSeparator                ;
Path.DirectorySeparatorChar       \
Directory.GetCurrentDirectory()   C:\cs11dotnet7\Chapter09\WorkingWithFileSystems
Environment.CurrentDirectory      C:\cs11dotnet7\Chapter09\WorkingWithFileSystems
Environment.SystemDirectory       C:\WINDOWS\system32
Path.GetTempPath()                C:\Users\markj\AppData\Local\Temp\
GetFolderPath(SpecialFolder
 .System)                         C:\WINDOWS\system32
 .ApplicationData)                C:\Users\markj\AppData\Roaming
 .MyDocuments)                    C:\Users\markj\OneDrive\Documents
 .Personal)                       C:\Users\markj\OneDrive\Documents
```

Good Practice: Windows uses a backslash `\` for the directory separator character. macOS and Linux use a forward slash `/` for the directory separator character. Do not assume what character is used in your code when combining paths.

Managing drives

To manage drives, use the `DriveInfo` type, which has a static method that returns information about all the drives connected to your computer. Each drive has a drive type.

Let's explore drives:

1. In `Program.cs`, write statements to get all the drives and output their name, type, size, available free space, and format, but only if the drive is ready, as shown in the following code:

```
SectionTitle("Managing drives");
WriteLine("{0,-30} | {1,-10} | {2,-7} | {3,18} | {4,18}",
  "NAME", "TYPE", "FORMAT", "SIZE (BYTES)", "FREE SPACE");
foreach (DriveInfo drive in DriveInfo.GetDrives())
{
  if (drive.IsReady)
  {
    WriteLine(
      "{0,-30} | {1,-10} | {2,-7} | {3,18:N0} | {4,18:N0}",
      drive.Name, drive.DriveType, drive.DriveFormat,
      drive.TotalSize, drive.AvailableFreeSpace);
  }
  else
  {
    WriteLine("{0,-30} | {1,-10}", drive.Name, drive.DriveType);
  }
}
```

Good Practice: Check that a drive is ready before reading properties such as `TotalSize` or you will see an exception thrown with removable drives.

1. Run the code and view the result, as shown in *Figure 9.2*:

Figure 9.2: Showing drive information on Windows

Managing directories

To manage directories, use the `Directory`, `Path`, and `Environment` static classes. These types include many members for working with the filesystem.

When constructing custom paths, you must be careful to write your code so that it makes no assumptions about the platform; for example, what to use for the directory separator character:

1. In `Program.cs`, write statements to do the following:
   - Define a custom path under the user's home directory by creating an array of strings for the directory names, and then properly combining them with the `Path` type's `Combine` method.
   - Check for the existence of the custom directory path using the `Exists` method of the `Directory` class.
   - Create and then delete the directory, including files and subdirectories within it, using the `CreateDirectory` and `Delete` methods of the `Directory` class:

```
SectionTitle("Managing directories");
// define a directory path for a new folder
// starting in the user's folder
string newFolder = Combine(
  GetFolderPath(SpecialFolder.Personal), "NewFolder");
WriteLine($"Working with: {newFolder}");
// check if it exists
WriteLine($"Does it exist? {Path.Exists(newFolder)}");
// create directory
WriteLine("Creating it...");
CreateDirectory(newFolder);
WriteLine($"Does it exist? {Path.Exists(newFolder)}");
Write("Confirm the directory exists, and then press ENTER: ");
ReadLine();
// delete directory
WriteLine("Deleting it...");
Delete(newFolder, recursive: true);
WriteLine($"Does it exist? {Path.Exists(newFolder)}");
```

1. Run the code and view the result, and use your favorite file management tool to confirm that the directory has been created before pressing Enter to delete it, as shown in the following output:

```
Working with: C:\Users\markj\OneDrive\Documents\NewFolder
Does it exist? False
Creating it...
Does it exist? True
Confirm the directory exists, and then press ENTER:
Deleting it...
Does it exist? False
```

Managing files

When working with files, you could statically import the file type, just as we did for the directory type, but, for the next example, we will not, because it has some of the same methods as the directory type and they would conflict. The file type has a short enough name not to matter in this case. The steps are as follows:

1. In `Program.cs`, write statements to do the following:
   - Check for the existence of a file.
   - Create a text file.
   - Write a line of text to the file.
   - Close the file to release system resources and file locks (this would normally be done inside a `try-finally` statement block to ensure that the file is closed even if an exception occurs when writing to it).
   - Copy the file to a backup.
   - Delete the original file.
   - Read the backup file's contents and then close it:

```
SectionTitle("Managing files");
// define a directory path to output files
// starting in the user's folder
string dir = Combine(
  GetFolderPath(SpecialFolder.Personal), "OutputFiles");
CreateDirectory(dir);
// define file paths
string textFile = Combine(dir, "Dummy.txt");
string backupFile = Combine(dir, "Dummy.bak");
WriteLine($"Working with: {textFile}");
// check if a file exists
WriteLine($"Does it exist? {File.Exists(textFile)}");
// create a new text file and write a line to it
StreamWriter textWriter = File.CreateText(textFile);
textWriter.WriteLine("Hello, C#!");
textWriter.Close(); // close file and release resources
WriteLine($"Does it exist? {File.Exists(textFile)}");
// copy the file, and overwrite if it already exists
File.Copy(sourceFileName: textFile,
```

```
  destFileName: backupFile, overwrite: true);
WriteLine(
  $"Does {backupFile} exist? {File.Exists(backupFile)}");
Write("Confirm the files exist, and then press ENTER: ");
ReadLine();
// delete file
File.Delete(textFile);
WriteLine($"Does it exist? {File.Exists(textFile)}");
// read from the text file backup
WriteLine($"Reading contents of {backupFile}:");
StreamReader textReader = File.OpenText(backupFile);
WriteLine(textReader.ReadToEnd());
textReader.Close();
```

1. Run the code and view the result, as shown in the following output:

```
Working with: C:\Users\markj\OneDrive\Documents\OutputFiles\Dummy.txt
Does it exist? False
Does it exist? True
Does C:\Users\markj\OneDrive\Documents\OutputFiles\Dummy.bak exist? True
Confirm the files exist, and then press ENTER:
Does it exist? False
Reading contents of C:\Users\markj\OneDrive\Documents\OutputFiles\Dummy.bak:
Hello, C#!
```

Managing paths

Sometimes, you need to work with parts of a path; for example, you might want to extract just the folder name, the filename, or the extension. Sometimes, you need to generate temporary folders and filenames. You can do this with static methods of the `Path` class:

1. In `Program.cs`, add the following statements:

```
SectionTitle("Managing paths");
WriteLine($"Folder Name: {GetDirectoryName(textFile)}");
WriteLine($"File Name: {GetFileName(textFile)}");
WriteLine("File Name without Extension: {0}",
  GetFileNameWithoutExtension(textFile));
WriteLine($"File Extension: {GetExtension(textFile)}");
WriteLine($"Random File Name: {GetRandomFileName()}");
WriteLine($"Temporary File Name: {GetTempFileName()}");
```

1. Run the code and view the result, as shown in the following output:

```
Folder Name: C:\Users\markj\OneDrive\Documents\OutputFiles
File Name: Dummy.txt
File Name without Extension: Dummy
File Extension: .txt
Random File Name: u45w1zki.co3
Temporary File Name:
/var/folders/tz/xx0y_wld5sx0nv0fjtq4tnpc0000gn/T/tmpyqrepP.tmp
```

`GetTempFileName` creates a zero-byte file and returns its name, ready for you to use. `GetRandomFileName` just returns a filename; it doesn't create the file.

Getting file information

To get more information about a file or directory, for example, its size or when it was last accessed, you can create an instance of the `FileInfo` or `DirectoryInfo` class.

`FileInfo` and `DirectoryInfo` both inherit from `FileSystemInfo`, so they both have members such as `LastAccessTime` and `Delete`, as well as extra members specific to themselves, as shown in the following table:

Class	Members

`FileSystemInfo`	Fields: FullPath, OriginalPath Properties: Attributes, CreationTime, CreationTimeUtc, Exists, Extension, FullName, LastAccessTime, LastAccessTimeUtc, LastWriteTime, LastWriteTimeUtc, Name Methods: Delete, GetObjectData, Refresh
`DirectoryInfo`	Properties: Parent, Root Methods: Create, CreateSubdirectory, EnumerateDirectories, EnumerateFiles, EnumerateFileSystemInfos, GetAccessControl, GetDirectories, GetFiles, GetFileSystemInfos, MoveTo, SetAccessControl
`FileInfo`	Properties: Directory, DirectoryName, IsReadOnly, Length Methods: AppendText, CopyTo, Create, CreateText, Decrypt, Encrypt, GetAccessControl, MoveTo, Open, OpenRead, OpenText, OpenWrite, Replace, SetAccessControl

Let's write some code that uses a `FileInfo` instance for efficiently performing multiple actions on a file:

1. In `Program.cs`, add statements to create an instance of `FileInfo` for the backup file and write information about it to the console, as shown in the following code:

```
SectionTitle("Getting file information");
FileInfo info = new(backupFile);
WriteLine($"{backupFile}:");
WriteLine($"Contains {info.Length} bytes");
WriteLine($"Last accessed {info.LastAccessTime}");
WriteLine($"Has readonly set to {info.IsReadOnly}");
```

1. Run the code and view the result, as shown in the following output:

```
C:\Users\markj\OneDrive\Documents\OutputFiles\Dummy.bak:
Contains 12 bytes
Last accessed 26/10/2022 09:08:26
Has readonly set to False
```

The number of bytes might be different on your operating system because operating systems can use different line endings.

Controlling how you work with files

When working with files, you often need to control how they are opened. The `File.Open` method has overloads to specify additional options using `enum` values.

The `enum` types are as follows:

- `FileMode`: This controls what you want to do with the file, like `CreateNew`, `OpenOrCreate`, or `Truncate`.
- `FileAccess`: This controls what level of access you need, like `ReadWrite`.
- `FileShare`: This controls locks on the file to allow other processes the specified level of access, like `Read`.

You might want to open a file and read from it, and allow other processes to read it too, as shown in the following code:

```
FileStream file = File.Open(pathToFile,
  FileMode.Open, FileAccess.Read, FileShare.Read);
```

There is also an `enum` for attributes of a file as follows:

- `FileAttributes` : This is to check a `FileSystemInfo` -derived type's `Attributes` property for values like `Archive` and `Encrypted`.

You could check a file or directory's attributes, as shown in the following code:

```
FileInfo info = new(backupFile);
WriteLine("Is the backup file compressed? {0}",
  info.Attributes.HasFlag(FileAttributes.Compressed));
```

Reading and writing with streams

A **stream** is a sequence of bytes that can be read from and written to. Although files can be processed rather like arrays, with random access provided by knowing the position of a byte within the file, it can be useful to process files as a stream in which the bytes can be accessed in sequential order.

Streams can also be used to process terminal input and output and networking resources such as sockets and ports that do not provide random access and cannot seek (that is, move) to a position. You can write code to process some arbitrary bytes without knowing or caring where they come from. Your code simply reads or writes to a stream, and another piece of code handles where the bytes are actually stored.

Understanding abstract and concrete streams

There is an `abstract` class named `Stream` that represents any type of stream. Remember that an `abstract` class cannot be instantiated using `new` ; it can only be inherited.

There are many concrete classes that inherit from this base class, including `FileStream` , `MemoryStream` , `BufferedStream` , `GZipStream` , and `SslStream` , so they all work the same way. All streams implement `IDisposable` , so they have a `Dispose` method to release unmanaged resources.

Some of the common members of the `Stream` class are described in the following table:

Member	Description
`CanRead` , `CanWrite`	These properties determine if you can read from and write to the stream.
`Length` , `Position`	These properties determine the total number of bytes and the current position within the stream. These properties may throw an exception for some types of streams.
`Dispose`	This method closes the stream and releases its resources.
`Flush`	If the stream has a buffer, then this method writes the bytes in the buffer to the stream and the buffer is cleared.
`CanSeek`	This property determines if the `Seek` method can be used.
`Seek`	This method moves the current position to the one specified in its parameter.
`Read` , `ReadAsync`	These methods read a specified number of bytes from the stream into a byte array and advance the position.
`ReadByte`	This method reads the next byte from the stream and advances the position.
`Write` , `WriteAsync`	These methods write the contents of a byte array into the stream.
`WriteByte`	This method writes a byte to the stream.

Understanding storage streams

Some storage streams that represent a location where the bytes will be stored are described in the following table:

Namespace	Class	Description
`System.IO`	`FileStream`	Bytes stored in the filesystem.
`System.IO`	`MemoryStream`	Bytes stored in memory in the current process.
`System.Net.Sockets`	`NetworkStream`	Bytes stored at a network location.

> `FileStream` has been re-written in .NET 6 to have much higher performance and reliability on Windows. You can read more about this at the following link: https://devblogs.microsoft.com/dotnet/file-io-improvements-in-dotnet-6/

Understanding function streams

Some function streams that cannot exist on their own, but can only be "plugged onto" other streams to add functionality, are described in the following table:

Namespace	Class	Description
`System.Security.Cryptography`	`CryptoStream`	This encrypts and decrypts the stream.
`System.IO.Compression`	`GZipStream` , `DeflateStream`	These compress and decompress the stream.
`System.Net.Security`	`AuthenticatedStream`	This sends credentials across the stream.

Understanding stream helpers

Although there will be occasions where you need to work with streams at a low level, most often, you can plug helper classes into the chain to make things easier. All the helper types for streams implement `IDisposable` , so they have a `Dispose` method to release unmanaged resources.

Some helper classes to handle common scenarios are described in the following table:

Namespace	Class	Description
`System.IO`	`StreamReader`	This reads from the underlying stream as plain text.
`System.IO`	`StreamWriter`	This writes to the underlying stream as plain text.
`System.IO`	`BinaryReader`	This reads from streams as .NET types. For example, the `ReadDecimal` method reads the next 16 bytes from the underlying stream as a `decimal` value and the `ReadInt32` method reads the next 4 bytes as an `int` value.
`System.IO`	`BinaryWriter`	This writes to streams as .NET types. For example, the `Write` method with a `decimal` parameter writes 16 bytes to the underlying stream and the `Write` method with an `int` parameter writes 4 bytes.
`System.Xml`	`XmlReader`	This reads from the underlying stream using XML format.
`System.Xml`	`XmlWriter`	This writes to the underlying stream using XML format.

Writing to text streams

Let's type some code to write text to a stream:

1. Use your preferred code editor to add a new **Console App** / `console` project named `WorkingWithStreams` to the `Chapter09` solution/workspace:
   - In Visual Studio, set the startup project for the solution to the current selection.
   - In Visual Studio Code, select `WorkingWithStreams` as the active OmniSharp project.
2. In the project file, add an element to statically and globally import the `System.Console` class.
3. Add a new class file named `Program.Helpers.cs` .

4. In `Program.Helpers.cs`, add a partial `Program` class with a `SectionTitle` method, as shown in the following code:

```
partial class Program
{
  static void SectionTitle(string title)
  {
    ConsoleColor previousColor = ForegroundColor;
    ForegroundColor = ConsoleColor.Yellow;
    WriteLine("*");
    WriteLine($"* {title}");
    WriteLine("*");
    ForegroundColor = previousColor;
  }
}
```

1. Add a new class file named `Viper.cs`.
2. In `Viper.cs`, define a static class named `Viper` with a static array of `string` values named `Callsigns`, as shown in the following code:

```
static class Viper
{
  // define an array of Viper pilot call signs
  public static string[] Callsigns = new[]
  {
    "Husker", "Starbuck", "Apollo", "Boomer",
    "Bulldog", "Athena", "Helo", "Racetrack"
  };
}
```

1. In `Program.cs`, delete the existing statements and then import the `System.Xml` namespace and statically import the `System.Environment` and `System.IO.Path` types.
2. In `Program.cs`, add statements to enumerate the Viper call signs, writing each one on its own line in a single text file, as shown in the following code:

```
SectionTitle("Writing to text streams");
// define a file to write to
string textFile = Combine(CurrentDirectory, "streams.txt");
// create a text file and return a helper writer
StreamWriter text = File.CreateText(textFile);
// enumerate the strings, writing each one
// to the stream on a separate line
foreach (string item in Viper.Callsigns)
{
  text.WriteLine(item);
}
text.Close(); // release resources
// output the contents of the file
WriteLine("{0} contains {1:N0} bytes.",
  arg0: textFile,
  arg1: new FileInfo(textFile).Length);
WriteLine(File.ReadAllText(textFile));
```

1. Run the code and view the result, as shown in the following output:

```
C:\cs11dotnet7\Chapter09\WorkingWithStreams\bin\Debug\net7.0\streams.txt contains 68 bytes.
Husker
Starbuck
Apollo
Boomer
Bulldog
Athena
Helo
Racetrack
```

1. Open the file that was created and check that it contains the list of call signs.

Writing to XML streams

There are two ways to write an XML element, as follows:

- `WriteStartElement` and `WriteEndElement` : Use this pair when an element might have child elements.
- `WriteElementString` : Use this when an element does not have children.

When you open a file to read or write to it, you are using resources outside of .NET. These are called **unmanaged resources** and must be disposed of when you are done working with them.

To deterministically control when they are disposed of, we can call the `Dispose` method inside a `finally` block.

Now, let's try storing the Viper pilot call signs array of `string` values in an XML file:

1. In `Program.cs` , add statements that enumerate the call signs, writing each one as an element in a single XML file, as shown in the following code:

```
SectionTitle("Writing to XML streams");
// define a file path to write to
string xmlFile = Combine(CurrentDirectory, "streams.xml");
// declare variables for the filestream and XML writer
FileStream? xmlFileStream = null;
XmlWriter? xml = null;
try
{
  // create a file stream
  xmlFileStream = File.Create(xmlFile);
  // wrap the file stream in an XML writer helper
  // and automatically indent nested elements
  xml = XmlWriter.Create(xmlFileStream,
    new XmlWriterSettings { Indent = true });
  // write the XML declaration
  xml.WriteStartDocument();
  // write a root element
  xml.WriteStartElement("callsigns");
  // enumerate the strings, writing each one to the stream
  foreach (string item in Viper.Callsigns)
  {
    xml.WriteElementString("callsign", item);
  }
  // write the close root element
  xml.WriteEndElement();
  // close helper and stream
  xml.Close();
  xmlFileStream.Close();
}
catch (Exception ex)
{
  // if the path doesn't exist the exception will be caught
  WriteLine($"{ex.GetType()} says {ex.Message}");
}
finally
{
  if (xml != null)
  {
    xml.Dispose();
    WriteLine("The XML writer's unmanaged resources have been disposed.");
    if (xmlFileStream != null)
    {
      xmlFileStream.Dispose();
      WriteLine("The file stream's unmanaged resources have been disposed.");
    }
  }
}
// output all the contents of the file
WriteLine("{0} contains {1:N0} bytes.",
  arg0: xmlFile,
```

```
  arg1: new FileInfo(xmlFile).Length);
WriteLine(File.ReadAllText(xmlFile));
```

1. Run the code and view the result, as shown in the following output:

```
The XML writer's unmanaged resources have been disposed.
The file stream's unmanaged resources have been disposed.
C:\cs11dotnet7\Chapter09\WorkingWithStreams\streams.xml contains
320 bytes.
<?xml version="1.0" encoding="utf-8"?>
<callsigns>
  <callsign>Husker</callsign>
  <callsign>Starbuck</callsign>
  <callsign>Apollo</callsign>
  <callsign>Boomer</callsign>
  <callsign>Bulldog</callsign>
  <callsign>Athena</callsign>
  <callsign>Helo</callsign>
  <callsign>Racetrack</callsign>
</callsigns>
```

Good Practice: Before calling the `Dispose` method, check that the object is not null.

Simplifying disposal by using the using statement

You can simplify the code that needs to check for a `null` object and then call its `Dispose` method by using the `using` statement. Generally, I would recommend using `using` rather than manually calling `Dispose` unless you need a greater level of control.

Confusingly, there are two uses for the `using` keyword: importing a namespace and generating a `finally` statement that calls `Dispose` on an object that implements `IDisposable`.

The compiler changes a `using` statement block into a `try-finally` statement without a `catch` statement. You can use nested `try` statements; so, if you do want to catch any exceptions, you can, as shown in the following code example:

```
using (FileStream file2 = File.OpenWrite(
  Path.Combine(path, "file2.txt")))
{
  using (StreamWriter writer2 = new StreamWriter(file2))
  {
    try
    {
      writer2.WriteLine("Welcome, .NET!");
    }
    catch(Exception ex)
    {
      WriteLine($"{ex.GetType()} says {ex.Message}");
    }
  } // automatically calls Dispose if the object is not null
} // automatically calls Dispose if the object is not null
```

You can even simplify the code further by not explicitly specifying the braces and indentation for the `using` statements, as shown in the following code:

```
using FileStream file2 = File.OpenWrite(
  Path.Combine(path, "file2.txt"));
using StreamWriter writer2 = new(file2);
try
{
  writer2.WriteLine("Welcome, .NET!");
}
catch(Exception ex)
{
  WriteLine($"{ex.GetType()} says {ex.Message}");
}
```

Compressing streams

XML is relatively verbose, so it takes up more space in bytes than plain text. Let's see how we can squeeze the XML using a common compression algorithm known as GZIP.

In .NET Core 2.1, Microsoft introduced an implementation of the Brotli compression algorithm. In performance, Brotli is like the algorithm used in DEFLATE and GZIP, but the output is about 20% denser.

Let's compare the two compression algorithms:

1. Add a new class file named `Program.Compress.cs`.
2. In `Program.Compress.cs`, write statements to use instances of `GZipStream` or `BrotliStream` to create a compressed file containing the same XML elements as before, and then decompress it while reading it and outputting to the console, as shown in the following code:

```
using System.IO.Compression; // BrotliStream, GZipStream, CompressionMode
using System.Xml; // XmlWriter, XmlReader
using static System.Environment; // CurrentDirectory
using static System.IO.Path; // Combine
partial class Program
{
  static void Compress(string algorithm = "gzip")
  {
    // define a file path using algorithm as file extension
    string filePath = Combine(
      CurrentDirectory, $"streams.{algorithm}");
    FileStream file = File.Create(filePath);
    Stream compressor;
    if (algorithm == "gzip")
    {
      compressor = new GZipStream(file, CompressionMode.Compress);
    }
    else
    {
      compressor = new BrotliStream(file, CompressionMode.Compress);
    }
    using (compressor)
    {
      using (XmlWriter xml = XmlWriter.Create(compressor))
      {
        xml.WriteStartDocument();
        xml.WriteStartElement("callsigns");
        foreach (string item in Viper.Callsigns)
        {
          xml.WriteElementString("callsign", item);
        }
      }
    } // also closes the underlying stream
    // output all the contents of the compressed file
    WriteLine("{0} contains {1:N0} bytes.",
      filePath, new FileInfo(filePath).Length);
    WriteLine($"The compressed contents:");
    WriteLine(File.ReadAllText(filePath));
    // read a compressed file
    WriteLine("Reading the compressed XML file:");
    file = File.Open(filePath, FileMode.Open);
    Stream decompressor;
    if (algorithm == "gzip")
    {
      decompressor = new GZipStream(
        file, CompressionMode.Decompress);
    }
    else
    {
      decompressor = new BrotliStream(
        file, CompressionMode.Decompress);
    }
    using (decompressor)
```

```
    using (XmlReader reader = XmlReader.Create(decompressor))

    while (reader.Read())
    {
      // check if we are on an element node named callsign
      if ((reader.NodeType == XmlNodeType.Element)
        && (reader.Name == "callsign"))
      {
        reader.Read(); // move to the text inside element
        WriteLine($"{reader.Value}"); // read its value
      }
    }
  }
}
```

1. In `Program.cs`, add calls to `Compress` with parameters to use `gzip` and `brotli` algorithms, as shown highlighted in the following code:

```
SectionTitle("Compressing streams");
Compress(algorithm: "gzip");
Compress(algorithm: "brotli");
```

1. Run the code and compare the sizes of the XML file and the compressed XML file using `gzip` and `brotli` algorithms, as shown in the following edited output:

```
C:\cs11dotnet7\Chapter09\WorkingWithStreams\bin\Debug\net7.0\streams.gzip contains 151 bytes.
The compressed contents:
▼?
z?{??}En?BYjQqf~???????Bj^r~Jf^??RiI??????MrbNNqfz^1?i?QZ??Zd?↨@H♣?$—%?&gc?t,?????*????H?????t?&
C:\cs11dotnet7\Chapter09\WorkingWithStreams\bin\Debug\net7.0\streams.brotli contains 118 bytes.
The compressed contents:
??∟     vl?9?L'??w?????.??lt???k?♥?L♥?-I↨hQ☻?^~{}?}?a{∟n?4xG?eX??V?#?Fp?P??w>0→¶?W?U??{???02/???
```

The compressed files are less than half the size of the same XML without compression, which was 320 bytes.

Encoding and decoding text

Text characters can be represented in different ways. For example, the alphabet can be encoded using Morse code into a series of dots and dashes for transmission over a telegraph line.

In a similar way, text inside a computer is stored as bits (ones and zeros) representing a code point within a code space. Most code points represent a single character, but they can also have other meanings like formatting.

For example, ASCII has a code space with 128 code points. .NET uses a standard called **Unicode** to encode text internally. Unicode has more than one million code points.

Sometimes, you will need to move text outside .NET for use by systems that do not use Unicode or use a variation of Unicode, so it is important to learn how to convert between encodings.

The following table lists some alternative text encodings commonly used by computers:

Encoding	Description
ASCII	This encodes a limited range of characters using the lower seven bits of a byte.
UTF-8	This represents each Unicode code point as a sequence of one to four bytes.
UTF-7	This is designed to be more efficient over 7-bit channels than UTF-8 but it has security and robustness issues, so UTF-8 is recommended over UTF-7.
UTF-16	This represents each Unicode code point as a sequence of one or two 16-bit integers.
UTF-32	This represents each Unicode code point as a 32-bit integer and is therefore a fixed-length encoding unlike the other Unicode encodings, which are all variable-length encodings.
ANSI/ISO	This provides support for a variety of code pages that are used to support a specific language or

encodings group of languages.

Good Practice: In most cases today, UTF-8 is a good default, which is why it is literally the default encoding, that is, `Encoding.Default`. You should avoid using `Encoding.UTF7` because it is unsecure. Due to this the C# compiler will warn you when you try to use UTF-7. Of course, you might need to generate text using that encoding for compatibility with another system, so it needs to remain an option in .NET.

Encoding strings as byte arrays

Let's explore text encodings:

1. Use your preferred code editor to add a new **Console App** / `console` project named `WorkingWithEncodings` to the `Chapter09` solution/workspace.
   - In Visual Studio Code, select `WorkingWithEncodings` as the active OmniSharp project.
2. In the project file, add an element to statically and globally import the `System.Console` class.
3. In `Program.cs`, delete the existing statements, import the `System.Text` namespace, and then add statements to encode a `string` using an encoding chosen by the user, loop through each byte, and then decode it back into a `string` and output it, as shown in the following code:

```
using System.Text;
WriteLine("Encodings");
WriteLine("[1] ASCII");
WriteLine("[2] UTF-7");
WriteLine("[3] UTF-8");
WriteLine("[4] UTF-16 (Unicode)");
WriteLine("[5] UTF-32");
WriteLine("[6] Latin1");
WriteLine("[any other key] Default encoding");
WriteLine();
// choose an encoding
Write("Press a number to choose an encoding.");
ConsoleKey number = ReadKey(intercept: true).Key;
WriteLine(); WriteLine();
Encoding encoder = number switch
{
  ConsoleKey.D1 or ConsoleKey.NumPad1 => Encoding.ASCII,
  ConsoleKey.D2 or ConsoleKey.NumPad2 => Encoding.UTF7,
  ConsoleKey.D3 or ConsoleKey.NumPad3 => Encoding.UTF8,
  ConsoleKey.D4 or ConsoleKey.NumPad4 => Encoding.Unicode,
  ConsoleKey.D5 or ConsoleKey.NumPad5 => Encoding.UTF32,
  ConsoleKey.D6 or ConsoleKey.NumPad6 => Encoding.Latin1,
  _              => Encoding.Default
};
// define a string to encode
string message = "Café £4.39";
WriteLine($"Text to encode: {message}  Characters: {message.Length}");
// encode the string into a byte array
byte[] encoded = encoder.GetBytes(message);
// check how many bytes the encoding needed
WriteLine("{0} used {1:N0} bytes.",
  encoder.GetType().Name, encoded.Length);
WriteLine();
// enumerate each byte
WriteLine($"BYTE | HEX | CHAR");
foreach (byte b in encoded)
{
  WriteLine($"{b,4} | {b.ToString("X"),3} | {(char)b,4}");
}
// decode the byte array back into a string and display it
string decoded = encoder.GetString(encoded);
WriteLine(decoded);
```

1. Run the code, press 1 to choose ASCII, and note that when outputting the bytes, the pound sign (£) and accented e (é) cannot be represented in ASCII, so it uses a question mark instead.

```
Text to encode: Café £4.39  Characters: 10
ASCIIEncodingSealed used 10 bytes.
BYTE | HEX | CHAR
  67 |  43 |    C
  97 |  61 |    a
 102 |  66 |    f
  63 |  3F |    ?
  32 |  20 |
  63 |  3F |    ?
  52 |  34 |    4
  46 |  2E |    .
  51 |  33 |    3
  57 |  39 |    9
Caf? ?4.39
```

1. Rerun the code and press 3 to choose UTF-8. Note that UTF-8 requires two extra bytes for the two characters that need 2 bytes each (12 bytes instead of 10 bytes total) but it can encode and decode the é and £ characters.

```
Text to encode: Café £4.39  Characters: 10
UTF8EncodingSealed used 12 bytes.
BYTE | HEX | CHAR
  67 |  43 |    C
  97 |  61 |    a
 102 |  66 |    f
 195 |  C3 |    Ã
 169 |  A9 |    ©
  32 |  20 |
 194 |  C2 |    Â
 163 |  A3 |    £
  52 |  34 |    4
  46 |  2E |    .
  51 |  33 |    3
  57 |  39 |    9
Café £4.39
```

1. Rerun the code and press 4 to choose Unicode (UTF-16). Note that UTF-16 requires two bytes for every character, so 20 bytes in total, and it can encode and decode the é and £ characters. This encoding is used internally by .NET to store `char` and `string` values.

Encoding and decoding text in files

When using stream helper classes, such as `StreamReader` and `StreamWriter`, you can specify the encoding you want to use. As you write to the helper, the text will automatically be encoded, and as you read from the helper, the bytes will be automatically decoded.

To specify an encoding, pass the encoding as a second parameter to the helper type's constructor, as shown in the following code:

```
StreamReader reader = new(stream, Encoding.UTF8);
StreamWriter writer = new(stream, Encoding.UTF8);
```

> **Good Practice**: Often, you won't have the choice of which encoding to use, because you will be generating a file for use by another system. However, if you do, pick one that uses the least number of bytes, but can store every character you need.

Reading and writing with random access handles

With .NET 6 and later, there is a new API for working with files without needing a file stream.

First, you must get a handle to the file, as shown in the following code:

```
using Microsoft.Win32.SafeHandles; // SafeFileHandle
using System.Text; // Encoding
using SafeFileHandle handle =
```

```
  File.OpenHandle(path: "coffee.txt",
    mode: FileMode.OpenOrCreate,
    access: FileAccess.ReadWrite);
```

You can then write some text encoded as a byte array and then stored in read-only memory buffer to the file, as shown in the following code:

```
string message = "Café £4.39";
ReadOnlyMemory<byte> buffer = new(Encoding.UTF8.GetBytes(message));
await RandomAccess.WriteAsync(handle, buffer, fileOffset: 0);
```

To read from the file, get the length of the file, allocate a memory buffer for the contents using that length, and then read the file, as shown in the following code:

```
long length = RandomAccess.GetLength(handle);
Memory<byte> contentBytes = new(new byte[length]);
await RandomAccess.ReadAsync(handle, contentBytes, fileOffset: 0);
string content = Encoding.UTF8.GetString(contentBytes.ToArray());
WriteLine($"Content of file: {content}");
```

Serializing object graphs

An **object graph** is multiple objects that are related to each other either through a direct reference or indirectly through a chain of references.

Serialization is the process of converting a live object graph into a sequence of bytes using a specified format. **Deserialization** is the reverse process. You would do this to save the current state of a live object so that you can recreate it in the future. For example, saving the current state of a game so that you can continue at the same place tomorrow. Serialized objects are usually stored in a file or database.

There are dozens of formats you can specify, but the two most common ones are **eXtensible Markup Language** (**XML**) and **JavaScript Object Notation** (**JSON**).

> **Good Practice**: JSON is more compact and is best for web and mobile applications. XML is more verbose but is better supported in more legacy systems. Use JSON to minimize the size of serialized object graphs. JSON is also a good choice when sending object graphs to web applications and mobile applications because JSON is the native serialization format for JavaScript and mobile apps often make calls over limited bandwidth, so the number of bytes is important.

.NET has multiple classes that will serialize to and from XML and JSON. We will start by looking at `XmlSerializer` and `JsonSerializer`.

Serializing as XML

Let's start by looking at XML, probably the world's most used serialization format (for now). To show a typical example, we will define a custom class to store information about a person and then create an object graph using a list of `Person` instances with nesting:

1. Use your preferred code editor to add a new **Console App** / `console` project named `WorkingWithSerialization` to the `Chapter09` solution/workspace.
   - In Visual Studio Code, select `WorkingWithSerialization` as the active OmniSharp project.
2. Add a new class file named `Person.cs` to define a `Person` class with a `Salary` property that is `protected`, meaning it is only accessible to itself and derived classes. To populate the salary, the class has a constructor with a single parameter to set the initial salary, as shown in the following code:

```
namespace Packt.Shared;
public class Person
{
  public Person(decimal initialSalary)
  {
```

```
    Salary = initialSalary;
  }
  public string? FirstName { get; set; }
  public string? LastName { get; set; }
  public DateTime DateOfBirth { get; set; }
  public HashSet<Person>? Children { get; set; }
  protected decimal Salary { get; set; }
}
```

1. In `Program.cs`, delete the existing statements. Import namespaces for working with XML serialization and statically import the `Environment` and `Path` classes, as shown in the following code:

```
using System.Xml.Serialization; // XmlSerializer
using Packt.Shared; // Person
using static System.Environment;
using static System.IO.Path;
```

1. In `Program.cs`, add statements to create an object graph of `Person` instances, as shown in the following code:

```
// create an object graph
List<Person> people = new()
{
  new(30000M)
  {
    FirstName = "Alice",
    LastName = "Smith",
    DateOfBirth = new(year: 1974, month: 3, day: 14)
  },
  new(40000M)
  {
    FirstName = "Bob",
    LastName = "Jones",
    DateOfBirth = new(year: 1969, month: 11, day: 23)
  },
  new(20000M)
  {
    FirstName = "Charlie",
    LastName = "Cox",
    DateOfBirth = new(year: 1984, month: 5, day: 4),
    Children = new()
    {
      new(0M)
      {
        FirstName = "Sally",
        LastName = "Cox",
        DateOfBirth = new(year: 2012, month: 7, day: 12)
      }
    }
  }
};
// create object that will format a List of Persons as XML
XmlSerializer xs = new(type: people.GetType());
// create a file to write to
string path = Combine(CurrentDirectory, "people.xml");
using (FileStream stream = File.Create(path))
{
  // serialize the object graph to the stream
  xs.Serialize(stream, people);
}
WriteLine("Written {0:N0} bytes of XML to {1}",
  arg0: new FileInfo(path).Length,
  arg1: path);
WriteLine();
// Display the serialized object graph
WriteLine(File.ReadAllText(path));
```

1. Run the code, view the result, and note that an exception is thrown, as shown in the following output:

```
Unhandled Exception: System.InvalidOperationException: Packt.Shared.Person cannot be serialized b
```

1. In `Person.cs`, add a statement to define a parameterless constructor, as shown in the following code:

```
public Person() { }
```

The constructor does not need to do anything, but it must exist so that the `XmlSerializer` can call it to instantiate new `Person` instances during the deserialization process.

1. Run the code and view the result, and note that the object graph is serialized as XML elements like `<FirstName>Bob</FirstName>` and that the `Salary` property is not included because it is not a `public` property, as shown in the following output:

```
Written 656 bytes of XML to C:\cs11dotnet7\Chapter09\WorkingWithSerialization\bin\Debug\net7.0\pe
<?xml version="1.0" encoding="utf-8"?><ArrayOfPerson xmlns:xsi="http://www.w3.org/2001/XMLSchema-
```

Generating compact XML

We could make the XML more compact using attributes instead of elements for some fields:

1. In `Person.cs`, import the `System.Xml.Serialization` namespace so that you can decorate some properties with the `[XmlAttribute]` attribute, as shown in the following code:

```
using System.Xml.Serialization; // [XmlAttribute]
```

1. In `Person.cs`, decorate the first name, last name, and date of birth properties with the `[XmlAttribute]` attribute, and set a short name for each property, as shown highlighted in the following code:

```
[XmlAttribute("fname")]
public string? FirstName { get; set; }
[XmlAttribute("lname")]
public string? LastName { get; set; }
[XmlAttribute("dob")]
public DateTime DateOfBirth { get; set; }
```

1. Run the code and note that the size of the file has been reduced from 656 to 451 bytes, a space-saving of more than a third, by outputting property values as XML attributes, as shown in the following output:

```
Written 451 bytes of XML to C:\cs11dotnet7\Chapter09\WorkingWithSerialization\bin\Debug\net7.0\pe
<?xml version="1.0" encoding="utf-8"?><ArrayOfPerson xmlns:xsi="http://www.w3.org/2001/XMLSchema-
```

Deserializing XML files

Now let's try deserializing the XML file back into live objects in memory:

1. In `Program.cs`, add statements to open the XML file and then deserialize it, as shown in the following code:

```
WriteLine();
WriteLine("* Deserializing XML files");
using (FileStream xmlLoad = File.Open(path, FileMode.Open))
{
  // deserialize and cast the object graph into a List of Person
  List<Person>? loadedPeople =
    xs.Deserialize(xmlLoad) as List<Person>;
  if (loadedPeople is not null)
  {
    foreach (Person p in loadedPeople)
    {
      WriteLine("{0} has {1} children.",
        p.LastName, p.Children?.Count ?? 0);
    }
  }
}
```

1. Run the code and note that the people are loaded successfully from the XML file and then enumerated, as shown in the following output:

```
* Deserializing XML files
Smith has 0 children.
Jones has 0 children.
Cox has 1 children.
```

There are many other attributes that can be used to control the XML generated.

If you don't use any annotations, `XmlSerializer` performs a case-insensitive match using the property name when deserializing.

> **Good Practice**: When using `XmlSerializer`, remember that only the public fields and properties are included, and the type must have a parameterless constructor. You can customize the output with attributes.

Serializing with JSON

One of the most popular .NET libraries for working with the JSON serialization format is **Newtonsoft.Json**, known as **Json.NET**. It is mature and powerful. Let's see it in action:

1. In the `WorkingWithSerialization` project, add a package reference for the latest version of `Newtonsoft.Json`, as shown in the following markup:

```
<ItemGroup>
  <PackageReference Include="Newtonsoft.Json" Version="13.0.1" />
</ItemGroup>
```

1. Build the `WorkingWithSerialization` project to restore packages.
2. In `Program.cs`, add statements to create a text file and then serialize the people into the file as JSON, as shown in the following code:

```
// create a file to write to
string jsonPath = Combine(CurrentDirectory, "people.json");
using (StreamWriter jsonStream = File.CreateText(jsonPath))
{
  // create an object that will format as JSON
  Newtonsoft.Json.JsonSerializer jss = new();
  // serialize the object graph into a string
  jss.Serialize(jsonStream, people);
}
WriteLine();
WriteLine("Written {0:N0} bytes of JSON to: {1}",
  arg0: new FileInfo(jsonPath).Length,
  arg1: jsonPath);
// display the serialized object graph
WriteLine(File.ReadAllText(jsonPath));
```

1. Run the code and note that JSON requires less than half the number of bytes compared to XML with elements. It's even smaller than the XML file, which uses attributes, as shown in the following output:

```
Written 366 bytes of JSON to: C:\cs11dotnet7\Chapter09\WorkingWithSerialization\bin\Debug\net7.0\
[{"FirstName":"Alice","LastName":"Smith","DateOfBirth":"1974-03-14T00:00:00","Children":null},{"F
```

High-performance JSON processing

.NET Core 3.0 introduced a new namespace for working with JSON, `System.Text.Json`, which is optimized for performance by leveraging APIs like `Span<T>`.

Also, older libraries like Json.NET are implemented by reading UTF-16. It would be more performant to read and write JSON documents using UTF-8 because most network protocols, including HTTP, use UTF-8 and you can avoid transcoding UTF-8 to and from Json.NET's Unicode `string` values.

With the new API, Microsoft achieved between 1.3x and 5x improvement, depending on the scenario.

The original author of Json.NET, James Newton-King, joined Microsoft and has been working with them to develop their new JSON types. As he says in a comment discussing the new JSON APIs, "Json.NET isn't going away":

Figure 9.3: A comment by the original author of Json.NET

Let's see how to use the new JSON APIs to deserialize a JSON file:

1. In the `WorkingWithSerialization` project, at the top of `Program.cs`, import the new JSON class for performing serialization using an alias to avoid conflicting names with the Json.NET one we used before, as shown in the following code:

```
using FastJson = System.Text.Json.JsonSerializer;
```

1. In `Program.cs`, add statements to open the JSON file, deserialize it, and output the names and counts of the children of the people, as shown in the following code:

```
WriteLine();
WriteLine("* Deserializing JSON files");
using (FileStream jsonLoad = File.Open(jsonPath, FileMode.Open))
{
  // deserialize object graph into a List of Person
  List<Person>? loadedPeople =
    await FastJson.DeserializeAsync(utf8Json: jsonLoad,
      returnType: typeof(List<Person>)) as List<Person>;
  if (loadedPeople is not null)
  {
    foreach (Person p in loadedPeople)
    {
      WriteLine("{0} has {1} children.",
        p.LastName, p.Children?.Count ?? 0);
    }
  }
}
```

1. Run the code and view the result, as shown in the following output:

```
* Deserializing JSON files
Smith has 0 children.
Jones has 0 children.
Cox has 1 children.
```

Good Practice: Choose Json.NET for developer productivity and a large feature set, or `System.Text.Json` for performance. You can review a list of the differences at the following link: https://docs.microsoft.com/en-us/dotnet/standard/serialization/system-text-json-migrate-from-newtonsoft-how-to?pivots=dotnet-7-0#table-of-differences-between-newtonsoftjson-and-systemtextjson

Controlling JSON processing

There are many options for taking control of how JSON is processed, as shown in the following list:

- Including and excluding fields.
- Setting a casing policy.
- Selecting a case-sensitivity policy.
- Choosing between compact and prettified whitespace.

Let's see some in action:

1. Use your preferred code editor to add a new **Console App** / `console` project named `WorkingWithJson` to the `Chapter09` solution/workspace.
   - In Visual Studio Code, select `WorkingWithJson` as the active OmniSharp project.
2. In the `WorkingWithJson` project, in `Program.cs`, delete the existing statements, import the main namespace for working with high performance JSON, and then statically import the `System.Environment` and `System.IO.Path` types, as shown in the following code:

```
using System.Text.Json; // JsonSerializer
using static System.Environment;
using static System.IO.Path;
```

1. Add a new class file named `Book.cs`.
2. In `Book.cs`, define a class named `Book`, as shown in the following code:

```
using System.Text.Json.Serialization; // [JsonInclude]
public class Book
{
  // constructor to set non-nullable property
  public Book(string title)
  {
    Title = title;
  }
  // properties
  public string Title { get; set; }
  public string? Author { get; set; }
  // fields
  [JsonInclude] // include this field
  public DateTime PublishDate;
  [JsonInclude] // include this field
  public DateTimeOffset Created;
  public ushort Pages;
}
```

1. In `Program.cs`, add statements to create an instance of the `Book` class and serialize it to JSON, as shown in the following code:

```
Book mybook = new(title:
  "C# 11 and .NET 7 - Modern Cross-Platform Development Fundamentals")
{
  Author = "Mark J Price",
  PublishDate = new(year: 2022, month: 11, day: 8),
  Pages = 823,
  Created = DateTimeOffset.UtcNow,
};
JsonSerializerOptions options = new()
{
  IncludeFields = true, // includes all fields
  PropertyNameCaseInsensitive = true,
  WriteIndented = true,
  PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
};
string filePath = Combine(CurrentDirectory, "mybook.json");
using (Stream fileStream = File.Create(filePath))
{
  JsonSerializer.Serialize<Book>(
    utf8Json: fileStream, value: mybook, options);
}
WriteLine("Written {0:N0} bytes of JSON to {1}",
  arg0: new FileInfo(filePath).Length,
  arg1: filePath);
```

```
WriteLine();
// display the serialized object graph
WriteLine(File.ReadAllText(filePath));
```

1. Run the code and view the result, as shown in the following output:

```
Written 222 bytes of JSON to C:\cs11dotnet7\Chapter09\WorkingWithJson\bin\Debug\net7.0\mybook.js
{
  "title": "C# 11 and .NET 7 - Modern Cross-Platform Development Fundamentals",
  "author": "Mark J Price",
  "publishDate": "2022-11-08T00:00:00",
  "created": "2022-03-04T08:16:38.1823225+00:00",
  "pages": 823
}
```

Note the following:

- The JSON file is 222 bytes.
- The member names use camelCasing, for example, `publishDate`. This is best for subsequent processing in a browser with JavaScript.
- All fields are included due to the options set, including `pages`.
- JSON is prettified for easier human legibility.
- `DateTime` and `DateTimeOffset` values are stored as a single standard `string` format.

1. In `Program.cs`, when setting the `JsonSerializerOptions`, comment out the setting of casing policy, write indented, and include fields.
2. Run the code and view the result, as shown in the following output:

```
Written 183 bytes of JSON to C:\cs11dotnet7\Chapter09\WorkingWithJson\bin\Debug\net7.0\mybook.js
{"Title":"C# 11 and .NET 7 - Modern Cross-Platform Development Fundamentals","Author":"Mark J Pr
```

Note the following:

- The JSON file is 183 bytes, a more than 20% reduction.
- The member names use normal casing, for example, `PublishDate`.
- The `Pages` field is missing. The other fields are included due to the `[JsonInclude]` attribute on `PublishDate` and `Created` field.

New JSON extension methods for working with HTTP responses

In .NET 5, Microsoft added refinements to the types in the `System.Text.Json` namespace like extension methods for `HttpResponse`, which you will see in *Chapter 15*, *Building and Consuming Web Services*.

Migrating from Newtonsoft to new JSON

If you have existing code that uses the Newtonsoft Json.NET library and you want to migrate to the new `System.Text.Json` namespace, then Microsoft has specific documentation for that, which you will find at the following link:

https://docs.microsoft.com/en-us/dotnet/standard/serialization/system-text-json-migrate-from-newtonsoft-how-to

Practicing and exploring

Test your knowledge and understanding by answering some questions, getting some hands-on practice, and exploring this chapter's topics with more in-depth research.

Exercise 9.1 – Test your knowledge

Answer the following questions:

1. What is the difference between using the `File` class and the `FileInfo` class?
2. What is the difference between the `ReadByte` method and the `Read` method of a stream?
3. When would you use the `StringReader`, `TextReader`, and `StreamReader` classes?
4. What does the `DeflateStream` type do?
5. How many bytes per character does UTF-8 encoding use?
6. What is an object graph?
7. What is the best serialization format to choose for minimizing space requirements?
8. What is the best serialization format to choose for cross-platform compatibility?
9. Why is it bad to use a `string` value like `"\Code\Chapter01"` to represent a path, and what should you do instead?
10. Where can you find information about NuGet packages and their dependencies?

Exercise 9.2 – Practice serializing as XML

In the `Chapter09` solution/workspace, create a console app named `Ch09Ex02SerializingShapes` that creates a list of shapes, uses serialization to save it to the filesystem using XML, and then deserializes it back:

```
// create a list of Shapes to serialize
List<Shape> listOfShapes = new()
{
  new Circle { Colour = "Red", Radius = 2.5 },
  new Rectangle { Colour = "Blue", Height = 20.0, Width = 10.0 },
  new Circle { Colour = "Green", Radius = 8.0 },
  new Circle { Colour = "Purple", Radius = 12.3 },
  new Rectangle { Colour = "Blue", Height = 45.0, Width = 18.0 }
};
```

Shapes should have a read-only property named `Area` so that when you deserialize, you can output a list of shapes, including their areas, as shown here:

```
List<Shape> loadedShapesXml =
  serializerXml.Deserialize(fileXml) as List<Shape>;
foreach (Shape item in loadedShapesXml)
{
  WriteLine("{0} is {1} and has an area of {2:N2}",
    item.GetType().Name, item.Colour, item.Area);
}
```

This is what your output should look like when you run your console application:

```
Loading shapes from XML:
Circle is Red and has an area of 19.63
Rectangle is Blue and has an area of 200.00
Circle is Green and has an area of 201.06
Circle is Purple and has an area of 475.29
Rectangle is Blue and has an area of 810.00
```

Exercise 9.3 – Explore topics

Use the links on the following page to learn more detail about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-9---working-with-files-streams-and-serialization

Summary

In this chapter, you learned how to:

- Read from and write to text files.
- Read from and write to XML files.
- Compress and decompress files.

- Encode and decode text.
- Serialize an object graph into JSON and XML.
- Deserialize an object graph from JSON and XML.

In the next chapter, you will learn how to work with databases using Entity Framework Core.

10 Working with Data Using Entity Framework Core

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about reading and writing to relational data stores, such as SQLite and Microsoft SQL Server, by using the object-to-data store mapping technology named **Entity Framework Core** (**EF Core**).

This chapter will cover the following topics:

- Understanding modern databases
- Setting up EF Core
- Defining EF Core models
- Querying EF Core models
- Loading patterns with EF Core
- Manipulating data with EF Core
- Working with transactions
- Code First EF Core models

Understanding modern databases

Two of the most common places to store data are in a **Relational Database Management System** (**RDBMS**) such as Microsoft SQL Server, PostgreSQL, MySQL, and SQLite, or in a **NoSQL** database such as Microsoft Azure Cosmos DB, Redis, MongoDB, and Apache Cassandra.

Understanding legacy Entity Framework

Entity Framework (**EF**) was first released as part of .NET Framework 3.5 with Service Pack 1 back in late 2008. Since then, Entity Framework has evolved, as Microsoft has observed how programmers use an **object-relational mapping** (**ORM**) tool in the real world.

ORMs use a mapping definition to associate columns in tables to properties in classes. Then, a programmer can interact with objects of different types in a way that they are familiar with, instead of having to deal with knowing how to store the values in a relational table or another structure provided by a NoSQL data store.

The version of EF included with .NET Framework is **Entity Framework 6** (**EF6**). It is mature, stable, and supports an EDMX (XML file) way of defining the model as well as complex inheritance models, and a few other advanced features.

EF 6.3 and later have been extracted from .NET Framework as a separate package so it can be supported on .NET Core 3.0 and later. This enables existing projects like web applications and services to be ported and run cross-platform. However, EF6 should be considered a legacy technology because it has some limitations when running cross-platform and no new features will be added to it.

Using the legacy Entity Framework 6.3 or later

To use the legacy Entity Framework in a .NET Core 3.0 or later project, you must add a package reference to it in your project file, as shown in the following markup:

```
<PackageReference Include="EntityFramework" Version="6.4.4" />
```

> **Good Practice**: Only use legacy EF6 if you must; for example, when migrating a WPF app that uses it. This book is about modern cross-platform development so, in the rest of this chapter, I will only cover the modern Entity Framework Core. You will not need to reference the legacy EF6 package as shown above in the projects for this chapter.

Understanding Entity Framework Core

The truly cross-platform version, **EF Core**, is different from the legacy Entity Framework. Although EF Core has a similar name, you should be aware of how it varies from EF6. The latest EF Core is version 7, to match .NET 7.

EF Core 5 and later only support .NET 5 and later. EF Core 3.0 and later only work with platforms that support .NET Standard 2.1, meaning .NET Core 3.0 and later. It does not support .NET Standard 2.0 platforms like .NET Framework 4.8.

As well as traditional RDBMSs, EF Core supports modern cloud-based, nonrelational, schema-less data stores, such as Microsoft Azure Cosmos DB and MongoDB, sometimes with third-party providers.

EF Core has so many improvements that this chapter cannot cover them all. I will focus on the fundamentals that all .NET developers should know and some of the cooler new features.

There are two approaches to working with EF Core:

1. **Database First**: A database already exists, so you build a model that matches its structure and features.
2. **Code First**: No database exists, so you build a model and then use EF Core to create a database that matches its structure and features.

We will start by using EF Core with an existing database.

Creating a console app for working with EF Core

First, we will create a console app project for this chapter:

1. Use your preferred code editor to create a new project, as defined in the following list:
   - Project template: **Console App** / `console`
   - Project file and folder: `WorkingWithEFCore`
   - Workspace/solution file and folder: `Chapter10`

Using a sample relational database

To learn how to manage an RDBMS using .NET, it would be useful to have a sample one so that you can practice on one that has a medium complexity and a decent number of sample records. Microsoft offers several sample databases, most of which are too complex for our needs, so instead, we will use a database that was first created in the early 1990s known as **Northwind**.

Let's take a minute to look at a diagram of the Northwind database. You can use the following diagram to refer to as we write code and queries throughout this book:

Figure 10.1: The Northwind database tables and relationships

You will write code to work with the `Categories` and `Products` tables later in this chapter, and other tables in later chapters. But before we do, note that:

- Each category has a unique identifier, name, description, and picture.
- Each product has a unique identifier, name, unit price, units in stock, and other fields.
- Each product is associated with a category by storing the category's unique identifier.
- The relationship between `Categories` and `Products` is one-to-many, meaning each category can have zero or more products.

Using SQLite

SQLite is a small, cross-platform, self-contained RDBMS that is available in the public domain. It's the most common RDBMS for mobile platforms such as iOS (iPhone and iPad) and Android.

Setting up SQLite for Windows

On Windows, we need to add the folder for SQLite to the system path so it will be found when we enter commands at a command prompt or terminal:

1. Start your favorite browser and navigate to the following link: https://www.sqlite.org/download.xhtml.
2. Scroll down the page to the **Precompiled Binaries for Windows** section.

3. Click **sqlite-tools-win32-x86-3380000.zip**. Note that the file might have a higher version number after this book is published, as shown in the following screenshot:

Figure 10.2: Downloading SQLite for Windows

4. Extract the ZIP file into a folder named `C:\Sqlite\`.
5. In the Windows **Start** menu, navigate to **Settings**.
6. Search for `environment` and choose **Edit the system environment variables**. On non-English versions of Windows, please search for the equivalent word in your local language to find the setting.
7. Click the **Environment Variables** button.
8. In **System variables**, select **Path** in the list, and then click **Edit…**.
9. If `C:\SQLite` is not already in the path, then click **New**, enter `C:\Sqlite`, and press Enter.
10. Click **OK**, click **OK**, click **OK**, and then close **Settings**.

Setting up SQLite for macOS

SQLite is included in macOS in the `/usr/bin/` directory as a command-line application named `sqlite3`.

Setting up SQLite for other OSes

SQLite can be downloaded and installed for other OSes from the following link: https://www.sqlite.org/download.xhtml.

Creating the Northwind sample database for SQLite

Now we can create the Northwind sample database for SQLite using an SQL script:

1. If you have not previously cloned the GitHub repository for this book, then do so now using the following link: https://github.com/markjprice/cs11dotnet7.

2. Copy the script to create the Northwind database for SQLite from the following path in your local Git repository: `/sql-scripts/Northwind4SQLite.sql` into the `WorkingWithEFCore` folder.
3. Start a command line in the `WorkingWithEFCore` folder:
   - On Windows, start **File Explorer**, right-click the `WorkingWithEFCore` folder, and select **New Command Prompt at Folder** or **Open in Windows Terminal**.
   - On macOS, start **Finder**, right-click the `WorkingWithEFCore` folder, and select **New Terminal at Folder**.
4. Enter the command to execute the SQL script using SQLite and create the `Northwind.db` database, as shown in the following command:

```
sqlite3 Northwind.db -init Northwind4SQLite.sql
```

1. Be patient because this command might take a while to create the database structure. Eventually, you will see the SQLite command prompt, as shown in the following output:

```
-- Loading resources from Northwind4SQLite.sql
SQLite version 3.38.0 2022-02-22 18:58:40
Enter ".help" for usage hints.
sqlite>
```

1. To exit SQLite command mode:
   - On Windows, press Ctrl + C.
   - On macOS, press Ctrl + D.
2. Leave your terminal or command prompt window open because you will use it again soon.

Managing the Northwind sample database with SQLiteStudio

You can use a cross-platform graphical database manager named **SQLiteStudio** to easily manage SQLite databases:

1. Navigate to the following link, https://sqlitestudio.pl, and download and extract the application to your preferred location.
2. Start **SQLiteStudio**.
3. On the **Database** menu, choose **Add a database**.

4. In the **Database** dialog, in the **File** section, click on the yellow folder button to browse for an existing database file on the local computer, select the `Northwind.db` file in the `WorkingWithEFCore` folder, and then click **OK**, as shown in *Figure 10.3*:

Figure 10.3: Adding the Northwind.db database file to SQLiteStudio

5. Right-click on the **Northwind** database and choose **Connect to the database**. You will see the 10 tables that were created by the script. (The script for SQLite is simpler than the one for SQL Server; it does not create as many tables or other database objects.)
6. Right-click on the **Products** table and choose **Edit the table**.

7. In the table editor window, note the structure of the `Products` table, including column names, data types, keys, and constraints, as shown in *Figure 10.4*:

Figure 10.4: The table editor in SQLiteStudio showing the structure of the Products table

8. In the table editor window, click the **Data** tab, and you will see 77 products, as shown in *Figure 10.5*:

Figure 10.5: The Data tab showing the rows in the Products table

9. In the **Database** window, right-click **Northwind** and select **Disconnect from the database**.
10. Exit SQLiteStudio.

Using the lightweight ADO.NET provider for SQLite

Before Entity Framework, there was **ADO.NET**. This is a simpler and more efficient API for working with databases. It provides abstract classes like `DbConnection`, `DbCommand`, and `DbReader`, and provider-specific implementations of them like `SqliteConnection` and `SqliteCommand`.

The Entity Framework Core provider for SQLite is built on top of this library, but it can also be used independently for better performance. This book does not cover using this library, but you can learn more about it at the following link:

https://docs.microsoft.com/en-us/dotnet/standard/data/sqlite/

Using Microsoft SQL Server for Windows

If you are using Windows and would prefer to use Microsoft SQL Server, please see the online instructions at the following link:

https://github.com/markjprice/cs11dotnet7/blob/main/docs/sql-server/README.md

Setting up EF Core

Before we dive into the practicalities of managing data using EF Core, let's briefly talk about choosing between EF Core data providers.

Choosing an EF Core database provider

To manage data in a specific database, we need classes that know how to efficiently talk to that database.

EF Core database providers are sets of classes that are optimized for a specific data store. There is even a provider for storing the data in the memory of the current process, which can be useful for high-performance unit testing since it avoids hitting an external system.

They are distributed as NuGet packages, as shown in the following table:

To manage this data store	Install this NuGet package
Microsoft SQL Server 2012 or later	`Microsoft.EntityFrameworkCore.SqlServer`
SQLite 3.7 or later	`Microsoft.EntityFrameworkCore.SQLite`
In-memory	`Microsoft.EntityFrameworkCore.InMemory`
Azure Cosmos DB SQL API	`Microsoft.EntityFrameworkCore.Cosmos`
MySQL	`MySQL.EntityFrameworkCore`
Oracle DB 11.2	`Oracle.EntityFrameworkCore`
PostgreSQL	`Npgsql.EntityFrameworkCore.PostgreSQL`

You can install as many EF Core database providers in the same project as you need. Each package includes the shared types as well as provider-specific types.

Connecting to a database

To connect to a SQLite database, we just need to know the database filename, set using the parameter `Filename`. We specify this information in a **connection string**.

Defining the Northwind database context class

The `Northwind` class will be used to represent the database. To use EF Core, the class must inherit from `DbContext`. This class understands how to communicate with databases and dynamically generate SQL statements to query and manipulate data.

Your `DbContext`-derived class should have an overridden method named `OnConfiguring`, which will set the database connection string.

We will create a project that uses SQLite, but feel free to use SQL Server instead:

1. In the `WorkingWithEFCore` project, add a package reference to the EF Core data provider for SQLite and globally and statically import the `System.Console` class for all C# files, as shown in the following markup:

```
<ItemGroup>
    <Using Include="System.Console" Static="true" />
</ItemGroup>
<ItemGroup>
  <PackageReference
```

```
    Include="Microsoft.EntityFrameworkCore.Sqlite"
    Version="7.0.0" />
</ItemGroup>
```

Early Access Program: Until the final release of .NET 7, the version numbers of packages will be for previews and release candidates. For example, `7.0.0-preview.1.22109.13`. You can change the version to `7.0.0-*` to automatically upgrade to the latest packages as monthly previews and release candidates are released.

1. Build the `WorkingWithEFCore` project to restore packages.
2. Add a new class file named `Northwind.cs`.
3. In `Northwind.cs`, define a class named `Northwind`, import the main namespace for EF Core, make the class inherit from `DbContext`, and in an `OnConfiguring` method, configure the options builder use SQLite, as shown in the following code:

```
using Microsoft.EntityFrameworkCore; // DbContext, DbContextOptionsBuilder
namespace Packt.Shared;
// this manages the connection to the database
public class Northwind : DbContext
{
  protected override void OnConfiguring(
    DbContextOptionsBuilder optionsBuilder)
  {
    string path = Path.Combine(
      Environment.CurrentDirectory, "Northwind.db");
    string connection = $"Filename={path}";
    ConsoleColor previousColor = ForegroundColor;
    ForegroundColor = ConsoleColor.DarkYellow;
    WriteLine($"Connection: {connection}");
    ForegroundColor = previousColor;
    optionsBuilder.UseSqlite(connection);
  }
}
```

Warning! If you are using Visual Studio for Windows, then the compiled application executes in the `WorkingWithEFCore\bin\Debug\net7.0` folder so it will not find the database file. If you are using Visual Studio Code, then the compiled application executes in the `WorkingWithEFCore` folder so it will find the database file.

1. In **Solution Explorer**, right-click the `Northwind.db` file and select **Properties**.
2. In **Properties**, set **Copy to Output Directory** to **Copy always**.
3. In `WorkingWithEFCore.csproj`, note the new elements, as shown in the following markup:

```
<ItemGroup>
  <None Update="Northwind.db">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
</ItemGroup>
```

1. In `Program.cs`, delete the existing statements and then import the `Packt.Shared` namespace and output the database provider, as shown in the following code:

```
using Packt.Shared;
Northwind db = new();
WriteLine($"Provider: {db.Database.ProviderName}");
```

1. Run the console app and note the output showing the database connection string and which database provider you are using, as shown in the following output:

```
Connection: Filename=C:\cs11dotnet7\Chapter10\WorkingWithEFCore\bin\Debug\net7.0\Northwind.db
Provider: Microsoft.EntityFrameworkCore.Sqlite
```

Defining EF Core models

EF Core uses a combination of **conventions**, **annotation attributes**, and **Fluent API** statements to build an entity model at runtime so that any actions performed on the classes can later be automatically translated into actions performed on the actual database. An entity class represents the structure of a table, and an instance of the class represents a row in that table.

First, we will review the three ways to define a model, with code examples, and then we will create some classes that implement those techniques.

Using EF Core conventions to define the model

The code we will write will use the following conventions:

- The name of a table is assumed to match the name of a `DbSet<T>` property in the `DbContext` class, for example, `Products`.
- The names of the columns are assumed to match the names of properties in the entity model class, for example, `ProductId`.
- The `string` .NET type is assumed to be a `nvarchar` type in the database.
- The `int` .NET type is assumed to be an `int` type in the database.
- The primary key is assumed to be a property that is named `Id` or `ID`, or when the entity model class is named `Product`, then the property can be named `ProductId` or `ProductID`. If this property is an integer type or the `Guid` type, then it is also assumed to be an `IDENTITY` column (a column type that automatically assigns a value when inserting).

Good Practice: There are many other conventions that you should know, and you can even define your own, but that is beyond the scope of this book. You can read about them at the following link: https://docs.microsoft.com/en-us/ef/core/modeling/

Using EF Core annotation attributes to define the model

Conventions often aren't enough to completely map the classes to the database objects. A simple way of adding more smarts to your model is to apply annotation attributes.

Some common attributes are shown in the following table:

Attribute	Description
`[Required]`	Ensures the value is not null.
`[StringLength(50)]`	Ensures the value is up to 50 characters in length.
`[RegularExpression(expression)]`	Ensures the value matches the specified regular expression.
`[Column(TypeName = "money", Name = "UnitPrice")]`	Specifies the column type and column name used in the table.

For example, in the database, the maximum length of a product name is 40, and the value cannot be null, as shown highlighted in the following **Data Definition Language** (**DDL**) code that defines how to create a table named `Products` with its columns, data types, keys, and other constraints:

```
CREATE TABLE Products (
    ProductId       INTEGER       PRIMARY KEY,
    ProductName     NVARCHAR (40) NOT NULL,
    SupplierId      "INT",
    CategoryId      "INT",
    QuantityPerUnit NVARCHAR (20),
    UnitPrice       "MONEY"       CONSTRAINT DF_Products_UnitPrice DEFAULT (0),
    UnitsInStock    "SMALLINT"    CONSTRAINT DF_Products_UnitsInStock DEFAULT (0),
    UnitsOnOrder    "SMALLINT"    CONSTRAINT DF_Products_UnitsOnOrder DEFAULT (0),
    ReorderLevel    "SMALLINT"    CONSTRAINT DF_Products_ReorderLevel DEFAULT (0),
    Discontinued    "BIT"         NOT NULL
                                  CONSTRAINT DF_Products_Discontinued DEFAULT (0),
```

```
    CONSTRAINT FK_Products_Categories FOREIGN KEY (
        CategoryId
    )
    REFERENCES Categories (CategoryId),
    CONSTRAINT FK_Products_Suppliers FOREIGN KEY (
        SupplierId
    )
    REFERENCES Suppliers (SupplierId),
    CONSTRAINT CK_Products_UnitPrice CHECK (UnitPrice >= 0),
    CONSTRAINT CK_ReorderLevel CHECK (ReorderLevel >= 0),
    CONSTRAINT CK_UnitsInStock CHECK (UnitsInStock >= 0),
    CONSTRAINT CK_UnitsOnOrder CHECK (UnitsOnOrder >= 0)
);
```

In a `Product` class, we could apply attributes to specify this, as shown in the following code:

```
[Required]
[StringLength(40)]
public string ProductName { get; set; }
```

When there isn't an obvious map between .NET types and database types, an attribute can be used.

For example, in the database, the column type of `UnitPrice` for the `Products` table is `money`. .NET does not have a `money` type, so it should use `decimal` instead, as shown in the following code:

```
[Column(TypeName = "money")]
public decimal? UnitPrice { get; set; }
```

Using the EF Core Fluent API to define the model

The last way that the model can be defined is by using the Fluent API. This API can be used instead of attributes, as well as being used in addition to them. For example, to define the `ProductName` property, instead of decorating the property with two attributes, an equivalent Fluent API statement could be written in the `OnModelCreating` method of the database context class, as shown in the following code:

```
modelBuilder.Entity<Product>()
  .Property(product => product.ProductName)
  .IsRequired()
  .HasMaxLength(40);
```

This keeps the entity model class simpler.

Understanding data seeding with the Fluent API

Another benefit of the Fluent API is to provide initial data to populate a database. EF Core automatically works out what insert, update, or delete operations must be executed.

For example, if we wanted to make sure that a new database has at least one row in the `Product` table, then we would call the `HasData` method, as shown in the following code:

```
modelBuilder.Entity<Product>()
  .HasData(new Product
  {
    ProductId = 1,
    ProductName = "Chai",
    UnitPrice = 8.99M
  });
```

Our model will map to an existing database that is already populated with data, so we will not need to use this technique in our code.

Building EF Core models for the Northwind tables

Now that you've learned about ways to define EF Core models, let's build models to represent two of the tables in the `Northwind` database.

The two entity classes will refer to each other, so to avoid compiler errors, we will create the classes without any members first:

1. In the `WorkingWithEFCore` project, add two class files named `Category.cs` and `Product.cs`.
2. In `Category.cs`, define a class named `Category`, as shown in the following code:

```
namespace Packt.Shared;
public class Category
{
}
```

1. In `Product.cs`, define a class named `Product`, as shown in the following code:

```
namespace Packt.Shared;
public class Product
{
}
```

Defining the Category and Product entity classes

The `Category` class, also known as an entity model, will be used to represent a row in the `Categories` table. This table has four columns, as shown in the following DDL:

```
CREATE TABLE Categories (
    CategoryId   INTEGER       PRIMARY KEY,
    CategoryName NVARCHAR (15) NOT NULL,
    Description  "NTEXT",
    Picture      "IMAGE"
);
```

We will use conventions to define:

- Three of the four properties (we will not map the `Picture` column).
- The primary key.
- The one-to-many relationship to the `Products` table.

To map the `Description` column to the correct database type, we will need to decorate the `string` property with the `Column` attribute.

Later in this chapter, we will use the Fluent API to define that `CategoryName` cannot be null and is limited to a maximum of 15 characters.

Let's go:

1. Modify the `Category` entity model class, as shown in the following code:

```
using System.ComponentModel.DataAnnotations.Schema; // [Column]
namespace Packt.Shared;
public class Category
{
  // these properties map to columns in the database
  public int CategoryId { get; set; }
  public string? CategoryName { get; set; }
  [Column(TypeName = "ntext")]
  public string? Description { get; set; }
  // defines a navigation property for related rows
  public virtual ICollection<Product> Products { get; set; }
  public Category()
  {
    // to enable developers to add products to a Category we must
```

```
    // initialize the navigation property to an empty collection
    Products = new HashSet<Product>();
  }
}
```

The `Product` class will be used to represent a row in the `Products` table, which has ten columns.

You do not need to include all columns from a table as properties of a class. We will only map six properties: `ProductId`, `ProductName`, `UnitPrice`, `UnitsInStock`, `Discontinued`, and `CategoryId`.

Columns that are not mapped to properties cannot be read or set using the class instances. If you use the class to create a new object, then the new row in the table will have `NULL` or some other default value for the unmapped column values in that row. You must make sure that those missing columns are optional or have default values set by the database or an exception will be thrown at runtime. In this scenario, the rows already have data values and I have decided that I do not need to read those values in this application.

We can rename a column by defining a property with a different name, like `Cost`, and then decorating the property with the `[Column]` attribute and specifying its column name, like `UnitPrice`.

The final property, `CategoryId`, is associated with a `Category` property that will be used to map each product to its parent category.

1. Modify the `Product` class, as shown in the following code:

```
using System.ComponentModel.DataAnnotations; // [Required], [StringLength]
using System.ComponentModel.DataAnnotations.Schema; // [Column]
namespace Packt.Shared;
public class Product
{
  public int ProductId { get; set; } // primary key
  [Required]
  [StringLength(40)]
  public string ProductName { get; set; } = null!;
  [Column("UnitPrice", TypeName = "money")]
  public decimal? Cost { get; set; } // property name != column name
  [Column("UnitsInStock")]
  public short? Stock { get; set; }
  public bool Discontinued { get; set; }
  // these two define the foreign key relationship
  // to the Categories table
  public int CategoryId { get; set; }
  public virtual Category Category { get; set; } = null!;
}
```

The two properties that relate the two entities, `Category.Products` and `Product.Category`, are both marked as `virtual`. This allows EF Core to inherit and override the properties to provide extra features, such as lazy loading.

Adding tables to the Northwind database context class

Inside your `DbContext`-derived class, you must define at least one property of the `DbSet<T>` type. These properties represent the tables. To tell EF Core what columns each table has, the `DbSet<T>` properties use generics to specify a class that represents a row in the table. That entity model class has properties that represent its columns.

The `DbContext`-derived class can optionally have an overridden method named `OnModelCreating`. This is where you can write Fluent API statements as an alternative to decorating your entity classes with attributes.

Let's write the code:

1. Modify the `Northwind` class to add statements to define two properties for the two tables and an `OnModelCreating` method, as shown highlighted in the following code:

```
public class Northwind : DbContext
{
  // these properties map to tables in the database
  public DbSet<Category>? Categories { get; set; }
  public DbSet<Product>? Products { get; set; }
  protected override void OnConfiguring(
    DbContextOptionsBuilder optionsBuilder)
  {
    ...
  }
  protected override void OnModelCreating(
    ModelBuilder modelBuilder)
  {
    // example of using Fluent API instead of attributes
    // to limit the length of a category name to 15
    modelBuilder.Entity<Category>()
      .Property(category => category.CategoryName)
      .IsRequired() // NOT NULL
      .HasMaxLength(15);
    if (Database.ProviderName?.Contains("Sqlite") ?? false)
    {
      // added to "fix" the lack of decimal support in SQLite
      modelBuilder.Entity<Product>()
        .Property(product => product.Cost)
        .HasConversion<double>();
    }
  }
}
```

In EF Core 3.0 and later, the `decimal` type is not supported by the SQLite database provider for sorting and other operations. We can fix this by telling the model that `decimal` values can be converted to `double` values when using the SQLite database provider. This does not actually perform any conversion at runtime.

Now that you have seen some examples of defining an entity model manually, let's see a tool that can do some of the work for you.

Setting up the dotnet-ef tool

The .NET CLI tool named `dotnet` can be extended with capabilities useful for working with EF Core. It can perform design-time tasks like creating and applying migrations from an older model to a newer model and generating code for a model from an existing database.

The `dotnet ef` command-line tool is not automatically installed. You must install this package as either a **global** or **local tool**. If you have already installed an older version of the tool, then you should uninstall any existing version:

1. At a command prompt or terminal, check if you have already installed `dotnet-ef` as a global tool, as shown in the following command:

```
dotnet tool list --global
```

1. Check in the list if an older version of the tool has been installed, like the one for .NET Core 3.1, as shown in the following output:

```
Package Id      Version     Commands
-------------------------------------
dotnet-ef       3.1.0       dotnet-ef
```

1. If an old version is already installed, then uninstall the tool, as shown in the following command:

```
dotnet tool uninstall --global dotnet-ef
```

1. Install the latest version, as shown in the following command:

```
dotnet tool install --global dotnet-ef --version 7.0-*
```

If necessary, follow any OS-specific instructions to add the `dotnet tools` directory to your `PATH` environment variable, as described in the output of installing the `dotnet-ef` tool.

Scaffolding models using an existing database

Scaffolding is the process of using a tool to create classes that represent the model of an existing database using reverse engineering. A good scaffolding tool allows you to extend the automatically generated classes and then regenerate those classes without losing your extended classes.

If you know that you will never regenerate the classes using the tool, then feel free to change the code for the automatically generated classes as much as you want. The code generated by the tool is just the best approximation.

Good Practice: Do not be afraid to overrule a tool when you know better.

Let's see if the tool generates the same model as we did manually:

1. Add the `Microsoft.EntityFrameworkCore.Design` package to the `WorkingWithEFCore` project.
2. At a command prompt or terminal in the `WorkingWithEFCore` folder, generate a model for the `Categories` and `Products` tables in a new folder named `AutoGenModels`, as shown in the following command:

```
dotnet ef dbcontext scaffold "Filename=Northwind.db" Microsoft.EntityFrameworkCore.Sqlite --table
```

Note the following:

- The command action: `dbcontext scaffold`
- The connection string: `"Filename=Northwind.db"`
- The database provider: `Microsoft.EntityFrameworkCore.Sqlite`
- The tables to generate models for: `--table Categories --table Products`
- The output folder: `--output-dir AutoGenModels`
- The namespace: `--namespace WorkingWithEFCore.AutoGen`
- To use data annotations as well as the Fluent API: `--data-annotations`
- To rename the context from `[database_name]Context`: `--context Northwind`

1. Note the build messages and warnings, as shown in the following output:

```
Build started...
Build succeeded.
To protect potentially sensitive information in your connection string, you should move it out o
Skipping foreign key with identity '0' on table 'Products' since principal table 'Suppliers' was
```

1. Open the `AutoGenModels` folder and note the three class files that were automatically generated: `Category.cs`, `Northwind.cs`, and `Product.cs`.
2. Open `Category.cs` and note the differences compared to the one you created manually, as shown in the following code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.EntityFrameworkCore;
namespace WorkingWithEFCore.AutoGen
{
  [Index("CategoryName", Name = "CategoryName")]
  public partial class Category
  {
    public Category()
    {
      Products = new HashSet<Product>();
    }
    [Key]
    public long CategoryId { get; set; }
```

```
    [Column(TypeName = "nvarchar (15)")]
    public string CategoryName { get; set; }
    [Column(TypeName = "ntext")]
    public string? Description { get; set; }
    [Column(TypeName = "image")]
    public byte[]? Picture { get; set; }
    [InverseProperty("Category")]
    public virtual ICollection<Product> Products { get; set; }
  }
}
```

Note the following:

- It decorates the entity class with the `[Index]` attribute that was introduced in EF Core 5.0. This indicates properties that should have an index. In earlier versions, only the Fluent API was supported for defining indexes. Since we are working with an existing database, this is not needed. But if we wanted to recreate a new empty database from our code, then this information would be needed.
- The table name in the database is `Categories` but the `dotnet-ef` tool uses the **Humanizer** third-party library to automatically singularize the class name to `Category`, which is a more natural name when creating a single entity.
- The entity class is declared using the `partial` keyword so that you can create a matching `partial` class for adding additional code. This allows you to rerun the tool and regenerate the entity class without losing that extra code.
- The `CategoryId` property is decorated with the `[Key]` attribute to indicate that it is the primary key for this entity. The data type for this property is `int` for SQL Server and `long` for SQLite.
- The `Products` property uses the `[InverseProperty]` attribute to define the foreign key relationship to the `Category` property on the `Product` entity class.

1. Open `Product.cs` and note the differences compared to the one you created manually.
2. Open `Northwind.cs` and note the differences compared to the one you created manually, as shown in the following edited-for-space code:

```
using Microsoft.EntityFrameworkCore;
namespace WorkingWithEFCore.AutoGen
{
  public partial class Northwind : DbContext
  {
    public Northwind()
    {
    }
    public Northwind(DbContextOptions<Northwind> options)
      : base(options)
    {
    }
    public virtual DbSet<Category> Categories { get; set; } = null!;
    public virtual DbSet<Product> Products { get; set; } = null!;
    protected override void OnConfiguring(
      DbContextOptionsBuilder optionsBuilder)
    {
      if (!optionsBuilder.IsConfigured)
      {
#warning To protect potentially sensitive information in your connection string, you should move
        optionsBuilder.UseSqlite("Filename=Northwind.db");
      }
    }
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
      modelBuilder.Entity<Category>(entity =>
      {
        ...
      });
      modelBuilder.Entity<Product>(entity =>
      {
        ...
      });
      OnModelCreatingPartial(modelBuilder);
```

```
    }
    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
  }
}
```

Note the following:

- The `Northwind` data context class is `partial` to allow you to extend it and regenerate it in the future.
- It has two constructors: a default parameter-less one and one that allows options to be passed in. This is useful in apps where you want to specify the connection string at runtime.
- The two `DbSet<T>` properties that represent the `Categories` and `Products` tables are set to the `null`-forgiving value to prevent static compiler analysis warnings at compile time. It has no effect at runtime.
- In the `OnConfiguring` method, if options have not been specified in the constructor, then it defaults to using a connection string that looks for the database file in the current folder. It has a compiler warning to remind you that you should not hardcode security information in this connection string.
- In the `OnModelCreating` method, the Fluent API is used to configure the two entity classes, and then a partial method named `OnModelCreatingPartial` is invoked. This allows you to implement that partial method in your own partial `Northwind` class to add your own Fluent API configuration that will not be lost if you regenerate the model classes.

1. Close the automatically generated class files.

Configuring preconvention models

Along with support for the `DateOnly` and `TimeOnly` types for use with the SQLite database provider, one of the new features introduced with EF Core 6 is configuring preconvention models.

As models become more complex, relying on conventions to discover entity types and their properties and successfully map them to tables and columns becomes harder. It would be useful if you could configure the conventions themselves before they are used to analyze and build a model.

For example, you might want to define a convention to say that all `string` properties should have a maximum length of 50 characters as a default, or any property types that implement a custom interface should not be mapped, as shown in the following code:

```
protected override void ConfigureConventions(
  ModelConfigurationBuilder configurationBuilder)
{
  configurationBuilder.Properties<string>().HaveMaxLength(50);
  configurationBuilder.IgnoreAny<IDoNotMap>();
}
```

In the rest of this chapter, we will use the classes that you manually created.

Querying EF Core models

Now that we have a model that maps to the Northwind database and two of its tables, we can write some simple LINQ queries to fetch data. You will learn much more about writing LINQ queries in *Chapter 11*, *Querying and Manipulating Data Using LINQ*.

For now, just write the code and view the results:

1. Add a new class file named `Program.Helpers.cs`.
2. In `Program.Helpers.cs`, add a partial `Program` class with a `SectionTitle` method, as shown in the following code:

```
partial class Program
{
  static void SectionTitle(string title)
  {
```

```
    ConsoleColor previousColor = ForegroundColor;
    ForegroundColor = ConsoleColor.Yellow;
    WriteLine("*");
    WriteLine($"* {title}");
    WriteLine("*");
    ForegroundColor = previousColor;
  }
  static void Fail(string message)
  {
    ConsoleColor previousColor = ForegroundColor;
    ForegroundColor = ConsoleColor.Red;
    WriteLine($"Fail > {message}");
    ForegroundColor = previousColor;
  }
  static void Info(string message)
  {
    ConsoleColor previousColor = ForegroundColor;
    ForegroundColor = ConsoleColor.Cyan;
    WriteLine($"Info > {message}");
    ForegroundColor = previousColor;
  }
}
```

1. Add a new class file named `Program.Queries.cs`.
2. In `Program.Queries.cs`, define a partial `Program` class with a `QueryingCategories` method, and add statements to do these tasks, as shown in the following code:
   - Create an instance of the `Northwind` class that will manage the database. Database context instances are designed for short lifetimes in a unit of work. They should be disposed of as soon as possible, so we will wrap it in a `using` statement. In *Chapter 13*, *Building Websites Using ASP.NET Core Razor Pages*, you will learn how to get a database context using dependency injection.
   - Create a query for all categories that includes their related products.
   - Enumerate through the categories, outputting the name and number of products for each one:

```
using Microsoft.EntityFrameworkCore; // Include extension method
using Packt.Shared; // Northwind, Category, Product
partial class Program
{
  static void QueryingCategories()
  {
    using (Northwind db = new())
    {
      SectionTitle("Categories and how many products they have:");
      // a query to get all categories and their related products
      IQueryable<Category>? categories = db.Categories?
        .Include(c => c.Products);
      if (categories is null)
      {
        Fail("No categories found.");
        return;
      }
      // execute query and enumerate results
      foreach (Category c in categories)
      {
        WriteLine($"{c.CategoryName} has {c.Products.Count} products.");
      }
    }
  }
}
```

1. In `Program.cs`, comment out the two statements that create a Northwind instance and output the database provider name, and then call the `QueryingCategories` method, as shown in the following code:

```
QueryingCategories();
```

1. Run the code and view the result (if run with Visual Studio 2022 for Windows using the SQLite database provider), as shown in the following partial output:

```
Beverages has 12 products.
Condiments has 12 products.
Confections has 13 products.
Dairy Products has 10 products.
Grains/Cereals has 7 products.
Meat/Poultry has 6 products.
Produce has 5 products.
Seafood has 12 products.
```

If you run with Visual Studio Code using the SQLite database provider, then the path will be the `WorkingWithEFCore` folder.

> **Warning!** If you see the following exception when using SQLite with Visual Studio 2022, the most likely problem is that the `Northwind.db` file is not being copied to the output directory. Make sure **Copy to Output Directory** is set to **Copy always**:

```
Unhandled exception. Microsoft.Data.Sqlite.SqliteException (0x80004005): SQLite Error 1: 'no such
```

Filtering included entities

EF Core 5.0 introduced **filtered includes**, which means you can specify a lambda expression in the `Include` method call to filter which entities are returned in the results:

1. In `Program.Queries.cs`, define a `FilteredIncludes` method, and add statements to do these tasks, as shown in the following code:
   - Create an instance of the `Northwind` class that will manage the database.
   - Prompt the user to enter a minimum value for units in stock.
   - Create a query for categories that have products with that minimum number of units in stock.
   - Enumerate through the categories and products, outputting the name and units in stock for each one:

```
static void FilteredIncludes()
{
  using (Northwind db = new())
  {
    SectionTitle("Products with a minimum number of units in stock.");
    string? input;
    int stock;
    do
    {
      Write("Enter a minimum for units in stock: ");
      input = ReadLine();
    } while (!int.TryParse(input, out stock));
    IQueryable<Category>? categories = db.Categories?
      .Include(c => c.Products.Where(p => p.Stock >= stock));
    if (categories is null)
    {
      Fail("No categories found.");
      return;
    }
    foreach (Category c in categories)
    {
      WriteLine($"{c.CategoryName} has {c.Products.Count} products with a minimum of {stock} unit
      foreach(Product p in c.Products)
      {
        WriteLine($"  {p.ProductName} has {p.Stock} units in stock.");
      }
    }
  }
}
```

1. In `Program.cs`, call the `FilteredIncludes` method, as shown in the following code:

```
FilteredIncludes();
```

1. Run the code, enter a minimum for units in stock like `100`, and view the result, as shown in the following partial output:

```
Enter a minimum for units in stock: 100
Beverages has 2 products with a minimum of 100 units in stock.
  Sasquatch Ale has 111 units in stock.
  Rhönbräu Klosterbier has 125 units in stock.
Condiments has 2 products with a minimum of 100 units in stock.
  Grandma's Boysenberry Spread has 120 units in stock.
  Sirop d'érable has 113 units in stock.
Confections has 0 products with a minimum of 100 units in stock.
Dairy Products has 1 products with a minimum of 100 units in stock.
  Geitost has 112 units in stock.
Grains/Cereals has 1 products with a minimum of 100 units in stock.
  Gustaf's Knäckebröd has 104 units in stock.
Meat/Poultry has 1 products with a minimum of 100 units in stock.
  Pâté chinois has 115 units in stock.
Produce has 0 products with a minimum of 100 units in stock.
Seafood has 3 products with a minimum of 100 units in stock.
  Inlagd Sill has 112 units in stock.
  Boston Crab Meat has 123 units in stock.
  Röd Kaviar has 101 units in stock.
```

Unicode characters in the Windows console

There is a limitation with the console provided by Microsoft on versions of Windows before the Windows 10 Fall Creators Update. By default, the console cannot display Unicode characters, for example, in the name Rhönbräu.

If you have this issue, then you can temporarily change the code page (also known as the character set) in a console to Unicode UTF-8 by entering the following command at the prompt before running the app:

```
chcp 65001
```

Filtering and sorting products

Let's explore a more complex query that will filter and sort data:

1. In `Program.Queries.cs`, define a `QueryingProducts` method, and add statements to do the following, as shown in the following code:
   - Create an instance of the `Northwind` class that will manage the database.
   - Prompt the user for a price for products. Unlike the previous code example, we will loop until the input is a valid price.
   - Create a query for products that cost more than the price using LINQ.
   - Loop through the results, outputting the ID, name, cost (formatted in US dollars), and the number of units in stock:

```
static void QueryingProducts()
{
  using (Northwind db = new())
  {
    SectionTitle("Products that cost more than a price, highest at top.");
    string? input;
    decimal price;
    do
    {
      Write("Enter a product price: ");
      input = ReadLine();
    } while (!decimal.TryParse(input, out price));
    IQueryable<Product>? products = db.Products?
      .Where(product => product.Cost > price)
      .OrderByDescending(product => product.Cost);
    if (products is null)
    {
      Fail("No products found.");
```

```
      return;
    }
    foreach (Product p in products)
    {
      WriteLine(
        "{0}: {1} costs {2:$#,##0.00} and has {3} in stock.",
        p.ProductId, p.ProductName, p.Cost, p.Stock);
    }
  }
}
```

1. In `Program.cs`, call the `QueryingProducts` method.
2. Run the code, enter `50` when prompted to enter a product price, and view the result, as shown in the following partial output:

```
Enter a product price: 50
38: Côte de Blaye costs $263.50 and has 17 in stock.
29: Thüringer Rostbratwurst costs $123.79 and has 0 in stock.
9: Mishi Kobe Niku costs $97.00 and has 29 in stock.
20: Sir Rodney's Marmalade costs $81.00 and has 40 in stock.
18: Carnarvon Tigers costs $62.50 and has 42 in stock.
59: Raclette Courdavault costs $55.00 and has 79 in stock.
51: Manjimup Dried Apples costs $53.00 and has 20 in stock.
```

Getting the generated SQL

You might be wondering how well written the SQL statements are that are generated from the C# queries we write. EF Core 5.0 introduced a quick and easy way to see the SQL generated:

1. In the `FilteredIncludes` method, before using the `foreach` statement to enumerate the query, add a statement to output the generated SQL, as shown in the following code:

```
Info($"ToQueryString: {categories.ToQueryString()}");
```

1. In the `QueryingProducts` method, before using the `foreach` statement to enumerate the query, add a statement to output the generated SQL, as shown in the following code:

```
Info($"ToQueryString: {products.ToQueryString()}");
```

1. Run the code, enter a minimum for units in stock like `99`, and view the result, as shown in the following partial output:

```
Enter a minimum for units in stock: 99
Connection: Filename=C:\cs11dotnet7\Chapter10\WorkingWithEFCore\bin\Debug\net7.0\Northwind.db
Info > ToQueryString: .param set @__stock_0 99
SELECT "c"."CategoryId", "c"."CategoryName", "c"."Description", "t"."ProductId", "t"."CategoryId"
FROM "Categories" AS "c"
LEFT JOIN (
    SELECT "p"."ProductId", "p"."CategoryId", "p"."UnitPrice", "p"."Discontinued", "p"."ProductNa
    FROM "Products" AS "p"
    WHERE "p"."UnitsInStock" >= @__stock_0
) AS "t" ON "c"."CategoryId" = "t"."CategoryId"
ORDER BY "c"."CategoryId"
Beverages has 2 products with a minimum of 99 units in stock.
  Sasquatch Ale has 111 units in stock.
  Rhönbräu Klosterbier has 125 units in stock.
...
```

Note the SQL parameter named `@__stock_0` has been set to a minimum stock value of `99`.

If you had used SQL Server, the generated SQL is slightly different. For example, it uses square brackets instead of double quotes around object names, as shown in the following output:

```
Info > ToQueryString: DECLARE @__stock_0 smallint = CAST(99 AS smallint);
SELECT [c].[CategoryId], [c].[CategoryName], [c].[Description], [t].[ProductId], [t].[CategoryId]
```

```
FROM [Categories] AS [c]
LEFT JOIN (
    SELECT [p].[ProductId], [p].[CategoryId], [p].[UnitPrice], [p].[Discontinued], [p].[ProductNa
    FROM [Products] AS [p]
    WHERE [p].[UnitsInStock] >= @__stock_0
) AS [t] ON [c].[CategoryId] = [t].[CategoryId]
ORDER BY [c].[CategoryId]
```

Logging EF Core

To monitor the interaction between EF Core and the database, we can enable logging. Logging could be to the console, to `Debug` or `Trace`, or to a file.

By default, EF Core logging will exclude any data in case it is sensitive. You can include this data by calling the `EnableSensitiveDataLogging` method, especially during development. You should disable it again before deploying to production.

Let's see an example of this in action:

1. In `Northwind.cs`, at the bottom of the `OnConfiguring` method, add a statement to log to the console, as shown in the following code:

```
optionsBuilder.LogTo(WriteLine) // Console
  .EnableSensitiveDataLogging();
```

   `LogTo` requires an `Action<string>` delegate. EF Core will call this delegate, passing a `string` value for each log message. Passing the `Console` class `WriteLine` method therefore tells the logger to write each method to the console.

1. Run the code and view the log messages, which are partially shown in the following output:

```
...
dbug: 05/03/2022 12:36:11.702 RelationalEventId.ConnectionOpening[20000] (Microsoft.EntityFramew
      Opening connection to database 'main' on server 'C:\cs11dotnet7\Chapter10\WorkingWithEFCore
dbug: 05/03/2022 12:36:11.718 RelationalEventId.ConnectionOpened[20001] (Microsoft.EntityFramewo
      Opened connection to database 'main' on server 'C:\cs11dotnet7\Chapter10\WorkingWithEFCore\
dbug: 05/03/2022 12:36:11.721 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT "c"."CategoryId", "c"."CategoryName", "c"."Description", "p"."ProductId", "p"."Cate
      FROM "Categories" AS "c"
      LEFT JOIN "Products" AS "p" ON "c"."CategoryId" = "p"."CategoryId"
      ORDER BY "c"."CategoryId"
...
```

Your logs might vary from those shown above based on your chosen database provider and code editor, and future improvements to EF Core. For now, note that different events like opening a connection or executing a command have different event IDs, as shown in the following list:

- 20000 `RelationalEventId.ConnectionOpening`: Includes the database file path.
- 20001 `RelationalEventId.ConnectionOpened`: Includes the database file path.
- 20100 `RelationalEventId.CommandExecuting`: Includes the SQL statement.

Filtering logs by provider-specific values

The event ID values and what they mean will be specific to the .NET data provider. If we want to know how the LINQ query has been translated into SQL statements and is executing, then the event ID to output has an `Id` value of `20100`:

1. Modify the `LogTo` method call to only output events with an `Id` of `20100`, as shown highlighted in the following code:

```
optionsBuilder.LogTo(WriteLine, // Console
  new[] { RelationalEventId.CommandExecuting })
  .EnableSensitiveDataLogging();
```

1. Run the code, and note the following SQL statements that were logged, as shown in the following output that has been edited for space:

```
dbug: 05/03/2022 12:48:43.153 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT "c"."CategoryId", "c"."CategoryName", "c"."Description", "p"."ProductId", "p"."Cate
      FROM "Categories" AS "c"
      LEFT JOIN "Products" AS "p" ON "c"."CategoryId" = "p"."CategoryId"
      ORDER BY "c"."CategoryId"
Beverages has 12 products.
Condiments has 12 products.
Confections has 13 products.
Dairy Products has 10 products.
Grains/Cereals has 7 products.
Meat/Poultry has 6 products.
Produce has 5 products.
Seafood has 12 products.
```

Logging with query tags

When logging LINQ queries, it can be tricky to correlate log messages in complex scenarios. EF Core 2.2 introduced the query tags feature to help by allowing you to add SQL comments to the log.

You can annotate a LINQ query using the `TagWith` method, as shown in the following code:

```
IQueryable<Product>? products = db.Products?
  .TagWith("Products filtered by price and sorted.")
  .Where(product => product.Cost > price)
  .OrderByDescending(product => product.Cost);
```

This will add a SQL comment to the log, as shown in the following output:

```
-- Products filtered by price and sorted.
```

Pattern matching with Like

EF Core supports common SQL statements including `Like` for pattern matching:

1. In `Program.Queries.cs`, add a method named `QueryingWithLike`, as shown in the following code, and note:
   - We have enabled logging.
   - We prompt the user to enter part of a product name and then use the `EF.Functions.Like` method to search anywhere in the `ProductName` property.
   - For each matching product, we output its name, stock, and if it is discontinued:

```
static void QueryingWithLike()
{
  using (Northwind db = new())
  {
    SectionTitle("Pattern matching with LIKE.");
    Write("Enter part of a product name: ");
    string? input = ReadLine();
    if (string.IsNullOrWhiteSpace(input))
    {
      Fail("You did not enter part of a product name.");
      return;
    }
    IQueryable<Product>? products = db.Products?
      .Where(p => EF.Functions.Like(p.ProductName, $"%{input}%"));
    if (products is null)
    {
```

```
      Fail("No products found.");
      return;
    }
    foreach (Product p in products)
    {
      WriteLine("{0} has {1} units in stock. Discontinued? {2}",
        p.ProductName, p.Stock, p.Discontinued);
    }
  }
}
```

1. In `Program.cs`, comment out the existing methods, and call `QueryingWithLike`.
2. Run the code, enter a partial product name such as `che`, and view the result, as shown in the following edited output:

```
Enter part of a product name: che
dbug: 05/03/2022 13:03:42.793 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[@__Format_1='%che%' (Size = 5)], CommandType='Text', Comma
      SELECT "p"."ProductId", "p"."CategoryId", "p"."UnitPrice", "p"."Discontinued", "p"."Product
      FROM "Products" AS "p"
      WHERE "p"."ProductName" LIKE @__Format_1
Chef Anton's Cajun Seasoning has 53 units in stock. Discontinued? False
Chef Anton's Gumbo Mix has 0 units in stock. Discontinued? True
Queso Manchego La Pastora has 86 units in stock. Discontinued? False
Gumbär Gummibärchen has 15 units in stock. Discontinued? False
```

Generating a random number in queries

EF Core 6.0 introduced a useful function, `EF.Functions.Random`, that maps to a database function returning a pseudo-random number between 0 and 1 exclusive. For example, you could multiply the random number by the count of rows in a table to select one random row from that table.

1. In `Program.Queries.cs`, add a method named `QueryingWithLike`, as shown in the following code:

```
static void GetRandomProduct()
{
  using (Northwind db = new())
  {
    SectionTitle("Get a random product.");
    int? rowCount = db.Products?.Count();
    if (rowCount == null)
    {
      Fail("Products table is empty.");
      return;
    }
    Product? p = db.Products?.FirstOrDefault(
      p => p.ProductId == (int)(EF.Functions.Random() * rowCount));
    if (p == null)
    {
      Fail("Product not found.");
      return;
    }
    WriteLine($"Random product: {p.ProductId} {p.ProductName}");
  }
}
```

1. In `Program.cs`, call `GetRandomProduct`.
2. Run the code and view the result, as shown in the following output:

```
dbug: 05/03/2022 13:19:01.783 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT COUNT(*)
      FROM "Products" AS "p"
dbug: 05/03/2022 13:19:01.848 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[@__p_1='77' (Nullable = true)], CommandType='Text', Comma
      SELECT "p"."ProductId", "p"."CategoryId", "p"."UnitPrice", "p"."Discontinued", "p"."Product
      FROM "Products" AS "p"
```

```
      WHERE "p"."ProductId" = CAST((abs(random() / 9.2233720368547799E+18) * @__p_1) AS INTEGER)
      LIMIT 1
Random product: 42 Singaporean Hokkien Fried Mee
```

Defining global filters

Northwind products can be discontinued, so it might be useful to ensure that discontinued products are never returned in results, even if the programmer does not use `Where` to filter them out in their queries:

1. In `Northwind.cs`, at the bottom of the `OnModelCreating` method, add a global filter to remove discontinued products, as shown in the following code:

```
// global filter to remove discontinued products
modelBuilder.Entity<Product>()
  .HasQueryFilter(p => !p.Discontinued);
```

1. In `Program.cs`, uncomment the call to `QueryingWithLike`, and comment out all the other method calls.
2. Run the code, enter the partial product name `che`, view the result, and note that **Chef Anton's Gumbo Mix** is now missing, because the SQL statement generated includes a filter for the `Discontinued` column, as shown highlighted in the following output:

```
Enter part of a product name: che
dbug: 05/03/2022 13:34:27.290 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[@__Format_1='%che%' (Size = 5)], CommandType='Text', Comma
      SELECT "p"."ProductId", "p"."CategoryId", "p"."UnitPrice", "p"."Discontinued", "p"."Product
      FROM "Products" AS "p"
      WHERE NOT ("p"."Discontinued") AND ("p"."ProductName" LIKE @__Format_1)
Chef Anton's Cajun Seasoning has 53 units in stock. Discontinued? False
Queso Manchego La Pastora has 86 units in stock. Discontinued? False
Gumbär Gummibärchen has 15 units in stock. Discontinued? False
```

Loading patterns with EF Core

There are three loading patterns that are commonly used with EF Core:

- **Eager loading**: Load data early.
- **Lazy loading**: Load data automatically just before it is needed.
- **Explicit loading**: Load data manually.

In this section, we're going to introduce each of them.

Eager loading entities

In the `QueryingCategories` method, the code currently uses the `Categories` property to loop through each category, outputting the category name and the number of products in that category.

This works because when we wrote the query, we enabled eager loading by calling the `Include` method for the related products.

Let's see what happens if we do not call `Include`:

1. Modify the query to comment out the `Include` method call, as shown in the following code:

```
IQueryable<Category>? categories = db.Categories;
  //.Include(c => c.Products);
```

1. In `Program.cs`, comment out all methods except `QueryingCategories`.
2. Run the code and view the result, as shown in the following partial output:

```
Beverages has 0 products.
Condiments has 0 products.
```

```
Confections has 0 products.
Dairy Products has 0 products.
Grains/Cereals has 0 products.
Meat/Poultry has 0 products.
Produce has 0 products.
Seafood has 0 products.
```

Each item in `foreach` is an instance of the `Category` class, which has a property named `Products`, that is, the list of products in that category. Since the original query is only selected from the `Categories` table, this property is empty for each category.

Enabling lazy loading

Lazy loading was introduced in EF Core 2.1, and it can automatically load missing related data. To enable lazy loading, developers must:

- Reference a NuGet package for proxies.
- Configure lazy loading to use a proxy.

Let's see this in action:

1. In the `WorkingWithEFCore` project, add a package reference for EF Core proxies, as shown in the following markup:

```
<PackageReference
  Include="Microsoft.EntityFrameworkCore.Proxies"
  Version="7.0.0" />
```

Early Access Program: Until the final release of .NET 7, the version numbers of packages will be for previews and release candidates. For example, `7.0.0-preview.1.22109.13`. You can change the version to `7.0.0-*` to automatically upgrade to the latest packages as monthly previews and release candidates are released.

1. Build the project to restore packages.
2. Open `Northwind.cs` and, at the bottom of the `OnConfiguring` method, call an extension method to use lazy loading proxies, as shown in the following code:

```
optionsBuilder.UseLazyLoadingProxies();
```

Now, every time the loop enumerates, and an attempt is made to read the `Products` property, the lazy loading proxy will check if they are loaded. If not, it will load them for us "lazily" by executing a `SELECT` statement to load just that set of products for the current category, and then the correct count will be returned to the output.

1. Run the code and note that the product counts are now correct. But you will see that the problem with lazy loading is that multiple round trips to the database server are required to eventually fetch all the data. For example, getting all the categories and then getting the products for the first category, `Beverages`, requires the execution of two SQL commands, as shown in the following partial output:

```
dbug: 05/03/2022 13:41:40.221 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT "c"."CategoryId", "c"."CategoryName", "c"."Description"
      FROM "Categories" AS "c"
dbug: 05/03/2022 13:41:40.331 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[@__p_0='1'], CommandType='Text', CommandTimeout='30']
      SELECT "p"."ProductId", "p"."CategoryId", "p"."UnitPrice", "p"."Discontinued", "p"."Produc
      FROM "Products" AS "p"
      WHERE NOT ("p"."Discontinued") AND "p"."CategoryId" = @__p_0
Beverages has 11 products.
```

Explicit loading entities

Another type of loading is explicit loading. It works in a similar way to lazy loading, with the difference being that you are in control of exactly what related data is loaded and when:

1. At the top of `Program.Queries.cs`, import the change tracking namespace to enable us to use the `CollectionEntry` class to manually load related entities, as shown in the following code:

```
using Microsoft.EntityFrameworkCore.ChangeTracking; // CollectionEntry
```

1. In `QueryingCategories`, modify the statements to disable lazy loading and then prompt the user as to whether they want to enable eager loading and explicit loading, as shown in the following code:

```
IQueryable<Category>? categories;
  // = db.Categories;
  // .Include(c => c.Products);
db.ChangeTracker.LazyLoadingEnabled = false;
Write("Enable eager loading? (Y/N): ");
bool eagerLoading = (ReadKey(intercept: true).Key == ConsoleKey.Y);
bool explicitLoading = false;
WriteLine();
if (eagerLoading)
{
  categories = db.Categories?.Include(c => c.Products);
}
else
{
  categories = db.Categories;
  Write("Enable explicit loading? (Y/N): ");
  explicitLoading = (ReadKey(intercept: true).Key == ConsoleKey.Y);
  WriteLine();
}
```

1. In the `foreach` loop, before the `WriteLine` method call, add statements to check if explicit loading is enabled, and if so, prompt the user as to whether they want to explicitly load each individual category, as shown in the following code:

```
if (explicitLoading)
{
  Write($"Explicitly load products for {c.CategoryName}? (Y/N): ");
  ConsoleKeyInfo key = ReadKey(intercept: true);
  WriteLine();
  if (key.Key == ConsoleKey.Y)
  {
    CollectionEntry<Category, Product> products =
      db.Entry(c).Collection(c2 => c2.Products);
    if (!products.IsLoaded) products.Load();
  }
}
```

1. Run the code:
   I. Press N to disable eager loading.
   II. Then press Y to enable explicit loading.
   III. For each category, press Y or N to load its products as you wish.

I chose to load products for only two of the eight categories, `Beverages` and `Seafood`, as shown in the following output that has been edited for space:

```
Enable eager loading? (Y/N):
Enable explicit loading? (Y/N):
dbug: 05/03/2022 13:48:48.541 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT "c"."CategoryId", "c"."CategoryName", "c"."Description"
      FROM "Categories" AS "c"
Explicitly load products for Beverages? (Y/N):
dbug: 05/03/2022 13:49:07.416 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[@__p_0='1'], CommandType='Text', CommandTimeout='30']
      SELECT "p"."ProductId", "p"."CategoryId", "p"."UnitPrice", "p"."Discontinued", "p"."Product
```

```
      FROM "Products" AS "p"
      WHERE NOT ("p"."Discontinued") AND "p"."CategoryId" = @__p_0
Beverages has 11 products.
Explicitly load products for Condiments? (Y/N):
Condiments has 0 products.
Explicitly load products for Confections? (Y/N):
Confections has 0 products.
Explicitly load products for Dairy Products? (Y/N):
Dairy Products has 0 products.
Explicitly load products for Grains/Cereals? (Y/N):
Grains/Cereals has 0 products.
Explicitly load products for Meat/Poultry? (Y/N):
Meat/Poultry has 0 products.
Explicitly load products for Produce? (Y/N):
Produce has 0 products.
Explicitly load products for Seafood? (Y/N):
dbug: 05/03/2022 13:49:16.682 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[@__p_0='8'], CommandType='Text', CommandTimeout='30']
      SELECT "p"."ProductId", "p"."CategoryId", "p"."UnitPrice", "p"."Discontinued", "p"."Produc
      FROM "Products" AS "p"
      WHERE NOT ("p"."Discontinued") AND "p"."CategoryId" = @__p_0
Seafood has 12 products.
```

Good Practice: Carefully consider which loading pattern is best for your code. Lazy loading could literally make you a lazy database developer! Read more about loading patterns at the following link: https://docs.microsoft.com/en-us/ef/core/querying/related-data

Manipulating data with EF Core

Inserting, updating, and deleting entities using EF Core is an easy task to accomplish.

`DbContext` maintains change tracking automatically, so the local entities can have multiple changes tracked, including adding new entities, modifying existing entities, and removing entities. When you are ready to send those changes to the underlying database, call the `SaveChanges` method. The number of entities successfully changed will be returned.

Inserting entities

Let's start by looking at how to add a new row to a table:

1. Add a new class file named `Program.Modifications.cs`.
2. In `Program.Modifications.cs`, create a partial `Program` class with a method named `ListProducts` that outputs the ID, name, cost, stock, and discontinued properties of each product sorted with the costliest first, as shown in the following code:

```
using Microsoft.EntityFrameworkCore.ChangeTracking; // EntityEntry<T>
using Packt.Shared;
partial class Program
{
  static void ListProducts(int? productIdToHighlight = null)
  {
    using (Northwind db = new())
    {
      if ((db.Products is null) || (db.Products.Count() == 0))
      {
        Fail("There are no products.");
        return;
      }
      WriteLine("| {0,-3} | {1,-35} | {2,8} | {3,5} | {4} |",
        "Id", "Product Name", "Cost", "Stock", "Disc.");
      foreach (Product p in db.Products)
      {
        ConsoleColor previousColor = ForegroundColor;
        if (productIdToHighlight == p.ProductId)
        {
```

```
          ForegroundColor = ConsoleColor.Yellow;
        }
        WriteLine("| {0:000} | {1,-35} | {2,8:$#,##0.00} | {3,5} | {4} |",
          p.ProductId, p.ProductName, p.Cost, p.Stock, p.Discontinued);
        ForegroundColor = previousColor;
      }
    }
  }
}
```

Remember that `1,-35` means left-align argument 1 within a 35-character-wide column and `3,5` means right-align argument 3 within a 5-character-wide column.

1. In `Program.Modifications.cs`, add a method named `AddProduct`, as shown in the following code:

```
static (int affected, int productId) AddProduct(
  int categoryId, string productName, decimal? price)
{
  using (Northwind db = new())
  {
    Product p = new()
    {
      CategoryId = categoryId,
      ProductName = productName,
      Cost = price,
      Stock = 72
    };
    // set product as added in change tracking
    EntityEntry<Product> entity = db.Products.Add(p);
    WriteLine($"State: {entity.State}, ProductId: {p.ProductId}");
    // save tracked change to database
    int affected = db.SaveChanges();
    WriteLine($"State: {entity.State}, ProductId: {p.ProductId}");
    return (affected, p.ProductId);
  }
}
```

1. In `Program.cs`, comment out previous method calls, and then call `AddProduct` and `ListProducts`, as shown in the following code:

```
var resultAdd = AddProduct(categoryId: 6,
  productName: "Bob's Burgers", price: 500M);
if (resultAdd.affected == 1)
{
  WriteLine("Add product successful.");
}
ListProducts(productIdToHighlight: resultAdd.productId);
```

1. Run the code, view the result, and note the new product has been added, as shown in the following partial output:

```
State: Added, ProductId: 0
dbug: 05/03/2022 14:21:37.818 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[@p0='6', @p1='500' (Nullable = true), @p2='False', @p3='B
      INSERT INTO "Products" ("CategoryId", "UnitPrice", "Discontinued", "ProductName", "UnitsIn
      VALUES (@p0, @p1, @p2, @p3, @p4);
      SELECT "ProductId"
      FROM "Products"
      WHERE changes() = 1 AND "rowid" = last_insert_rowid();
State: Unchanged, ProductId: 78
Add product successful.
| Id  | Product Name                        |     Cost | Stock | Disc. |
| 001 | Chai                                |   $18.00 |    39 | False |
| 002 | Chang                               |   $19.00 |    17 | False |
...
| 078 | Bob's Burgers                       |  $500.00 |    72 | False |
```

If you are using Visual Studio 2022 and SQLite, remember that we set the database file to copy to the output directory every time. This means it will be reset to 77 products each time you run the console app. If you use Visual Studio Code, the database file will retain any added rows in the product table.

Updating entities

Now, let's modify an existing row in a table:

1. In `Program.Modifications.cs`, add a method to increase the price of the first product with a name that begins with a specified value (we'll use `Bob` in our example) by a specified amount like `$20`, as shown in the following code:

```
static (int affected, int productId) IncreaseProductPrice(
  string productNameStartsWith, decimal amount)
{
  using (Northwind db = new())
  {
    // get first product whose name starts with name
    Product updateProduct = db.Products.First(
      p => p.ProductName.StartsWith(productNameStartsWith));
    updateProduct.Cost += amount;
    int affected = db.SaveChanges();
    return (affected, updateProduct.ProductId);
  }
}
```

1. In `Program.cs`, add statements to call `IncreaseProductPrice` and then `ListProducts`, as shown in the following code:

```
var resultUpdate = IncreaseProductPrice(
  productNameStartsWith: "Bob", amount: 20M);
if (resultUpdate.affected == 1)
{
  WriteLine("Increase product price successful.");
}
ListProducts(productIdToHighlight: resultUpdate.productId);
```

1. Run the code, view the result, and note that the existing entity for Bob's Burgers has increased in price by $20, as shown in the following partial output:

```
dbug: 05/03/2022 14:44:47.024 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[@__productNameStartsWith_0='Bob' (Size = 3)], CommandType=
      SELECT "p"."ProductId", "p"."CategoryId", "p"."UnitPrice", "p"."Discontinued", "p"."Product
      FROM "Products" AS "p"
      WHERE NOT ("p"."Discontinued") AND (@__productNameStartsWith_0 = '' OR (("p"."ProductName"
      LIMIT 1
dbug: 05/03/2022 14:44:47.028 RelationalEventId.CommandExecuting[20100] (Microsoft.EntityFramewo
      Executing DbCommand [Parameters=[@p1='78', @p0='520' (Nullable = true)], CommandType='Text
      UPDATE "Products" SET "UnitPrice" = @p0
      WHERE "ProductId" = @p1;
      SELECT changes();
Increase product price successful.
| Id  | Product Name                        |     Cost | Stock | Disc. |
| 001 | Chai                                |   $18.00 |    39 | False |
...
| 078 | Bob's Burgers                       |  $520.00 |    72 | False |
```

Deleting entities

You can remove individual entities with the `Remove` method. `RemoveRange` is more efficient when you want to delete multiple entities.

Let's see how to delete rows from a table:

1. In `Program.Modifications.cs`, add a method to delete all products with a name that begins with a specified value (`Bob` in our example), as shown in the following code:

```
static int DeleteProducts(string productNameStartsWith)
{
  using (Northwind db = new())
  {
    IQueryable<Product>? products = db.Products?.Where(
      p => p.ProductName.StartsWith(productNameStartsWith));
    if (products is null)
    {
      WriteLine("No products found to delete.");
      return 0;
    }
    else
    {
      db.Products.RemoveRange(products);
    }
    int affected = db.SaveChanges();
    return affected;
  }
}
```

1. In `Program.cs`, add statements to call `DeleteProducts`, as shown in the following code:

```
WriteLine("About to delete all products whose name starts with Bob.");
Write("Press Enter to continue: ");
if (ReadKey(intercept: true).Key == ConsoleKey.Enter)
{
  int deleted = DeleteProducts(productNameStartsWith: "Bob");
  WriteLine($"{deleted} product(s) were deleted.");
}
else
{
  WriteLine("Delete was cancelled.");
}
```

1. Run the code and view the result, as shown in the following partial output:

```
1 product(s) were deleted.
```

If multiple product names started with Bob, then they would all be deleted. As an optional challenge, modify the statements to add three new products that start with Bob and then delete them.

Pooling database contexts

The `DbContext` class is disposable and is designed following the single-unit-of-work principle. In the previous code examples, we created all the `DbContext`-derived Northwind instances in a `using` block so that `Dispose` is properly called at the end of each unit of work.

A feature of ASP.NET Core that is related to EF Core is that it makes your code more efficient by pooling database contexts when building websites and services. This allows you to create and dispose of as many `DbContext`-derived objects as you want, knowing that your code is still as efficient as possible.

Working with transactions

Every time you call the `SaveChanges` method, an **implicit transaction** is started so that if something goes wrong, it will automatically roll back all the changes. If the multiple changes within the transaction succeed, then the transaction and all changes are committed.

Transactions maintain the integrity of your database by applying locks to prevent reads and writes while a sequence of changes is occurring.

Transactions are **ACID**, which is an acronym explained in the following list:

- **A is for atomic**. Either all the operations in the transaction commit, or none of them do.
- **C is for consistent**. The state of the database before and after a transaction is consistent. This is dependent on your code logic; for example, when transferring money between bank accounts, it is up to your business logic to ensure that if you debit $100 in one account, you credit $100 in the other account.
- **I is for isolated**. During a transaction, changes are hidden from other processes. There are multiple isolation levels that you can pick from (refer to the following table). The stronger the level, the better the integrity of the data. However, more locks must be applied, which will negatively affect other processes. `Snapshot` is a special case because it creates multiple copies of rows to avoid locks, but this will increase the size of your database while transactions occur.
- **D is for durable**. If a failure occurs during a transaction, it can be recovered. This is often implemented as a two-phase commit and transaction logs. Once the transaction is committed it is guaranteed to endure even if there are subsequent errors. The opposite of durable is volatile.

Controlling transactions using isolation levels

A developer can control transactions by setting an **isolation level**, as described in the following table:

Isolation level	Lock(s)	Integrity problems allowed
`ReadUncommitted`	None	Dirty reads, non-repeatable reads, and phantom data
`ReadCommitted`	When editing, it applies read lock(s) to block other users from reading the record(s) until the transaction ends	Non-repeatable reads and phantom data
`RepeatableRead`	When reading, it applies edit lock(s) to block other users from editing the record(s) until the transaction ends	Phantom data
`Serializable`	Applies key-range locks to prevent any action that would affect the results, including inserts and deletes	None
`Snapshot`	None	None

Defining an explicit transaction

You can control explicit transactions using the `Database` property of the database context:

1. In `Program.Modifications.cs`, import the EF Core storage namespace to use the `IDbContextTransaction` interface, as shown in the following code:

```
using Microsoft.EntityFrameworkCore.Storage; // IDbContextTransaction
```

1. In the `DeleteProducts` method, after the instantiation of the `db` variable, add statements to start an explicit transaction and output its isolation level. At the bottom of the method, commit the transaction, and close the brace, as shown highlighted in the following code:

```
static int DeleteProducts(string productNameStartsWith)
{
  using (Northwind db = new())
  {
    using (IDbContextTransaction t = db.Database.BeginTransaction())
    {
      WriteLine("Transaction isolation level: {0}",
        arg0: t.GetDbTransaction().IsolationLevel);
      IQueryable<Product>? products = db.Products?.Where(
        p => p.ProductName.StartsWith(productNameStartsWith));
      if (products is null)
      {
        WriteLine("No products found to delete.");
        return 0;
      }
      else
```

```
      {
        db.Products.RemoveRange(products);
      }
      int affected = db.SaveChanges();
      t.Commit();
      return affected;
    }
  }
}
```

1. Run the code and view the result using SQLite, as shown in the following output:

```
Transaction isolation level: Serializable
```

If you were to use SQL Server, you would see the following output:

```
Transaction isolation level: ReadCommitted
```

Code First EF Core models

Sometimes you will not have an existing database. Instead, you define the EF Core model as Code First, and then EF Core can generate a matching database using create and drop APIs.

Good Practice: The create and drop APIs should only be used during development. Once you release the app, you do not want it to delete a production database!

For example, we might need to create an application for managing students and courses for an academy. One student can sign up to attend multiple courses. One course can be attended by multiple students. This is an example of a many-to-many relationship between students and courses.

Let's model this example:

1. Use your preferred code editor to add a new **Console App** / `console` project named `CoursesAndStudents` to the `Chapter10` solution/workspace.
   - In Visual Studio, set the startup project for the solution to the current selection.
   - In Visual Studio Code, select `CoursesAndStudents` as the active OmniSharp project.
2. In the `CoursesAndStudents` project, add package references for the following packages:
   - `Microsoft.EntityFrameworkCore.Sqlite`
   - `Microsoft.EntityFrameworkCore.Design`
3. Build the `CoursesAndStudents` project to restore packages.
4. Add classes named `Academy.cs`, `Student.cs`, and `Course.cs`.
5. Modify `Student.cs`, and note that it is a **POCO** (**plain old CLR object**) with no attributes decorating the class, as shown in the following code:

```
namespace Packt.Shared;
public class Student
{
  public int StudentId { get; set; }
  public string? FirstName { get; set; }
  public string? LastName { get; set; }
  public ICollection<Course>? Courses { get; set; }
}
```

1. Modify `Course.cs`, and note that we have decorated the `Title` property with some attributes to provide more information to the model, as shown in the following code:

```
using System.ComponentModel.DataAnnotations;
namespace Packt.Shared;
public class Course
{
  public int CourseId { get; set; }
  [Required]
  [StringLength(60)]
```

```
  public string? Title { get; set; }
  public ICollection<Student>? Students { get; set; }
}
```

1. Modify `Academy.cs`, as shown in the following code:

```
using Microsoft.EntityFrameworkCore; // DbContext, DbSet<T>
namespace Packt.Shared;
public class Academy : DbContext
{
  public DbSet<Student>? Students { get; set; }
  public DbSet<Course>? Courses { get; set; }
  protected override void OnConfiguring(
    DbContextOptionsBuilder optionsBuilder)
  {
    string path = Path.Combine(
      Environment.CurrentDirectory, "Academy.db");
    string connection = $"Filename={path}";
    // string connection = @"Data Source=.;Initial Catalog=Academy;Integrated Security=true;Mult:
    WriteLine($"Connection: {connection}");
    optionsBuilder.UseSqlite(connection);
    // optionsBuilder.UseSqlServer(connection);
  }
  protected override void OnModelCreating(ModelBuilder modelBuilder)
  {
    // Fluent API validation rules
    modelBuilder.Entity<Student>()
      .Property(s => s.LastName).HasMaxLength(30).IsRequired();
    // populate database with sample data
    Student alice = new() { StudentId = 1,
      FirstName = "Alice", LastName = "Jones" };
    Student bob = new() { StudentId = 2,
      FirstName = "Bob", LastName = "Smith" };
    Student cecilia = new() { StudentId = 3,
      FirstName = "Cecilia", LastName = "Ramirez" };
    Course csharp = new() { CourseId = 1,
      Title = "C# 11 and .NET 7"};
    Course webdev = new() { CourseId = 2,
      Title = "Web Development" };
    Course python = new() { CourseId = 3,
        Title = "Python for Beginners" };
    modelBuilder.Entity<Student>()
      .HasData(alice, bob, cecilia);
    modelBuilder.Entity<Course>()
      .HasData(csharp, webdev, python);
    modelBuilder.Entity<Course>()
      .HasMany(c => c.Students)
      .WithMany(s => s.Courses)
      .UsingEntity(e => e.HasData(
        // all students signed up for C# course
        new { CoursesCourseId = 1, StudentsStudentId = 1 },
        new { CoursesCourseId = 1, StudentsStudentId = 2 },
        new { CoursesCourseId = 1, StudentsStudentId = 3 },
        // only Bob signed up for Web Dev
        new { CoursesCourseId = 2, StudentsStudentId = 2 },
        // only Cecilia signed up for Python
        new { CoursesCourseId = 3, StudentsStudentId = 3 }
      ));
  }
}
```

Good Practice: Use an anonymous type to supply data for the intermediate table in a many-to-many relationship. The property names follow the naming convention `NavigationPropertyNamePropertyName`; for example, `Courses` is the navigation property name and `CourseId` is the property name so `CoursesCourseId` will be the property name of the anonymous type.

1. In `Program.cs`, delete the existing statements and then import the namespace for EF Core and working with tasks, and statically import `Console`, as shown in the following code:

```
using Microsoft.EntityFrameworkCore; // for GenerateCreateScript()
using Packt.Shared; // Academy
```

1. In `Program.cs`, add statements to create an instance of the `Academy` database context and use it to delete the database if it exists, create the database from the model and output the SQL script it uses, and then enumerate the students and their courses, as shown in the following code:

```
using (Academy a = new())
{
  bool deleted = await a.Database.EnsureDeletedAsync();
  WriteLine($"Database deleted: {deleted}");
  bool created = await a.Database.EnsureCreatedAsync();
  WriteLine($"Database created: {created}");
  WriteLine("SQL script used to create database:");
  WriteLine(a.Database.GenerateCreateScript());
  foreach (Student s in a.Students.Include(s => s.Courses))
  {
    WriteLine("{0} {1} attends the following {2} courses:",
      s.FirstName, s.LastName, s.Courses.Count);
    foreach (Course c in s.Courses)
    {
      WriteLine($"  {c.Title}");
    }
  }
}
```

1. Run the code, and note that the first time you run the code it will not need to delete the database because it does not exist yet, as shown in the following output:

```
Connection: Filename=C:\cs11dotnet7\Chapter10\CoursesAndStudents\bin\Debug\net7.0\Academy.db
Database deleted: False
Database created: True
SQL script used to create database:
CREATE TABLE "Courses" (
    "CourseId" INTEGER NOT NULL CONSTRAINT "PK_Courses" PRIMARY KEY AUTOINCREMENT,
    "Title" TEXT NOT NULL
);
CREATE TABLE "Students" (
    "StudentId" INTEGER NOT NULL CONSTRAINT "PK_Students" PRIMARY KEY AUTOINCREMENT,
    "FirstName" TEXT NULL,
    "LastName" TEXT NOT NULL
);
CREATE TABLE "CourseStudent" (
    "CoursesCourseId" INTEGER NOT NULL,
    "StudentsStudentId" INTEGER NOT NULL,
    CONSTRAINT "PK_CourseStudent" PRIMARY KEY ("CoursesCourseId", "StudentsStudentId"),
    CONSTRAINT "FK_CourseStudent_Courses_CoursesCourseId" FOREIGN KEY ("CoursesCourseId") REFEREI
    CONSTRAINT "FK_CourseStudent_Students_StudentsStudentId" FOREIGN KEY ("StudentsStudentId") RI
);
INSERT INTO "Courses" ("CourseId", "Title")
VALUES (1, 'C# 11 and .NET 7');
INSERT INTO "Courses" ("CourseId", "Title")
VALUES (2, 'Web Development');
INSERT INTO "Courses" ("CourseId", "Title")
VALUES (3, 'Python for Beginners');
INSERT INTO "Students" ("StudentId", "FirstName", "LastName")
VALUES (1, 'Alice', 'Jones');
INSERT INTO "Students" ("StudentId", "FirstName", "LastName")
VALUES (2, 'Bob', 'Smith');
INSERT INTO "Students" ("StudentId", "FirstName", "LastName")
VALUES (3, 'Cecilia', 'Ramirez');
INSERT INTO "CourseStudent" ("CoursesCourseId", "StudentsStudentId")
VALUES (1, 1);
INSERT INTO "CourseStudent" ("CoursesCourseId", "StudentsStudentId")
VALUES (1, 2);
INSERT INTO "CourseStudent" ("CoursesCourseId", "StudentsStudentId")
VALUES (1, 3);
INSERT INTO "CourseStudent" ("CoursesCourseId", "StudentsStudentId")
VALUES (2, 2);
INSERT INTO "CourseStudent" ("CoursesCourseId", "StudentsStudentId")
```

```
VALUES (3, 3);
CREATE INDEX "IX_CourseStudent_StudentsStudentId" ON "CourseStudent" ("StudentsStudentId");
Alice Jones attends the following 1 courses:
  C# 11 and .NET 7
Bob Smith attends the following 2 courses:
  C# 11 and .NET 7
  Web Development
Cecilia Ramirez attends the following 2 courses:
  C# 11 and .NET 7
  Python for Beginners
```

Note the following:

- The `Title` column is `NOT NULL` because the model was decorated with `[Required]`.
- The `LastName` column is `NOT NULL` because the model used `IsRequired()`.
- An intermediate table named `CourseStudent` was created to hold information about which students attend which courses.

1. Use SQLiteStudio to connect to the `Academy` database and view the tables, as shown in *Figure 10.6*:

Figure 10.6: Viewing the Academy database tables in SQLiteStudio

Understanding migrations

After publishing a project that uses a database, it is likely that you will later need to change your entity data model and therefore the database structure. At that point, you should not use the `Ensure` methods. Instead, you need to use a system that allows you to incrementally update the database schema while preserving any existing data in the database. EF Core migrations are that system.

Migrations get complex fast, so are beyond the scope of this book. You can read about them at the following link: https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/

Practicing and exploring

Test your knowledge and understanding by answering some questions, getting some hands-on practice, and exploring this chapter's topics with deeper research.

Exercise 10.1 – Test your knowledge

Answer the following questions:

1. What type would you use for the property that represents a table, for example, the `Products` property of a database context?
2. What type would you use for the property that represents a one-to-many relationship, for example, the `Products` property of a `Category` entity?
3. What is the EF Core convention for primary keys?
4. When might you use an annotation attribute in an entity class?
5. Why might you choose the Fluent API in preference to annotation attributes?
6. What does a transaction isolation level of `Serializable` mean?
7. What does the `DbContext.SaveChanges()` method return?
8. What is the difference between eager loading and explicit loading?
9. How should you define an EF Core entity class to match the following table?

```
CREATE TABLE Employees(
  EmpId INT IDENTITY,
  FirstName NVARCHAR(40) NOT NULL,
  Salary MONEY
)
```

1. What benefit do you get from declaring entity navigation properties as `virtual`?

Exercise 10.2 – Practice exporting data using different serialization formats

In the `Chapter10` solution/workspace, create a console app named `Ch10Ex02DataSerialization` that queries the Northwind database for all the categories and products, and then serializes the data using at least three formats of serialization available to .NET. Which format of serialization uses the least number of bytes?

Exercise 10.3 – Explore topics

Use the links on the following page to learn more detail about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-10---working-with-data-using-entity-framework-core

Exercise 10.4 – Explore NoSQL databases

This chapter focused on RDBMSs such as SQL Server and SQLite. If you wish to learn more about NoSQL databases, such as Cosmos DB and MongoDB, and how to use them with EF Core, then I recommend the following links:

- **Welcome to Azure Cosmos DB**: https://docs.microsoft.com/en-us/azure/cosmos-db/introduction

- **Use NoSQL databases as a persistence infrastructure**: https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/microservice-ddd-cqrs-patterns/nosql-database-persistence-infrastructure
- **Document Database Providers for Entity Framework Core**: https://github.com/BlueshiftSoftware/EntityFrameworkCore

Summary

In this chapter, you learned how to:

- Connect to and build entity data models for an existing database.
- Execute a simple LINQ query and process the results.
- Use filtered includes.
- Add, modify, and delete data.
- Define a Code First model and use it to create a new database and populate it with data.

In the next chapter, you will learn how to write more advanced LINQ queries to select, filter, sort, join, and group.

11 Querying and Manipulating Data Using LINQ

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about **Language INtegrated Query** (**LINQ**) expressions. LINQ is a set of language extensions that add the ability to work with sequences of data and then filter, sort, and project them into different outputs.

This chapter will cover the following topics:

- Writing LINQ expressions
- Working with sets using LINQ
- Using LINQ with EF Core
- Sweetening LINQ syntax with syntactic sugar
- Using multiple threads with parallel LINQ
- Creating your own LINQ extension methods
- Working with LINQ to XML

Writing LINQ expressions

Although we wrote a few LINQ expressions in *Chapter 10*, *Working with Data Using Entity Framework Core*, they weren't the focus, and so I didn't properly explain how LINQ works, so let's now take time to properly understand them.

What makes LINQ?

LINQ has several parts; some are required, and some are optional:

- **Extension methods (required)**: These include examples such as `Where`, `OrderBy`, and `Select`. These are what provide the functionality of LINQ.
- **LINQ providers (required)**: These include **LINQ to Objects** for processing in-memory objects, **LINQ to Entities** for processing data stored in external databases and modeled with EF Core, and **LINQ to XML** for processing data stored as XML. These providers are what execute LINQ expressions in a way specific to different types of data.
- **Lambda expressions (optional)**: These can be used instead of named methods to simplify LINQ queries, for example, for the conditional logic of the `Where` method for filtering.
- **LINQ query comprehension syntax (optional)**: These include C# keywords like `from`, `in`, `where`, `orderby`, `descending`, and `select`. These are aliases for some of the LINQ extension methods, and their use can simplify the queries you write, especially if you already have experience with other query languages, such as **Structured Query Language** (**SQL**).

When programmers are first introduced to LINQ, they often believe that LINQ query comprehension syntax is LINQ, but ironically, that is one of the parts of LINQ that is optional!

Building LINQ expressions with the Enumerable class

The LINQ extension methods, such as `Where` and `Select`, are appended by the `Enumerable` static class to any type, known as a **sequence**, that implements `IEnumerable<T>`.

For example, an array of any type implements the `IEnumerable<T>` class, where `T` is the type of item in the array. This means that all arrays support LINQ to query and manipulate them.

All generic collections, such as `List<T>`, `Dictionary<TKey, TValue>`, `Stack<T>`, and `Queue<T>`, implement `IEnumerable<T>`, so they can be queried and manipulated with LINQ too.

`Enumerable` defines more than 50 extension methods, as summarized in the following table:

Method(s)	Description
`First`, `FirstOrDefault`, `Last`, `LastOrDefault`	Get the first or last item in the sequence or throw an exception, or return the default value for the type, for example, `0` for an `int` and `null` for a reference type, if there is not a first or last item.
`Where`	Return a sequence of items that match a specified filter.
`Single`, `SingleOrDefault`	Return an item that matches a specific filter or throw an exception, or return the default value for the type if there is not exactly one match.
`ElementAt`, `ElementAtOrDefault`	Return an item at a specified index position or throw an exception, or return the default value for the type if there is not an item at that position. New in .NET 6 are overloads that can be passed an Index instead of an int, which is more efficient when working with `Span<T>` sequences.
`Select`, `SelectMany`	Project items into a different shape, that is, a different type, and flatten a nested hierarchy of items.
`OrderBy`, `OrderByDescending`, `ThenBy`, `ThenByDescending`	Sort items by a specified field or property.
`Reverse`	Reverse the order of the items.
`GroupBy`, `GroupJoin`, `Join`	Group and/or join two sequences.
`Skip`, `SkipWhile`	Skip a number of items; or skip while an expression is `true`.
`Take`, `TakeWhile`	Take a number of items; or take while an expression is `true`. Introduced in .NET 6 is an overload that can be passed a `Range`, for example, `Take(range: 3..^5)` meaning take a subset starting 3 items in from the start and ending 5 items in from the end, or instead of `Skip(4)` you could use `Take(4..)`.
`Aggregate`, `Average`, `Count`, `LongCount`, `Max`, `Min`, `Sum`	Calculate aggregate values.
`TryGetNonEnumeratedCount`	`Count()` checks if a `Count` property is implemented on the sequence and returns its value, or it enumerates the entire sequence to count its items. Introduced in .NET 6 is this method that only checks for `Count` and if it is missing it returns `false` and sets the `out` parameter to `0` to avoid a potentially poor-performing operation.
`All`, `Any`, `Contains`	Return `true` if all or any of the items match the filter, or if the sequence contains a specified item.
`Cast<T>`	Cast items into a specified type. It is useful to convert non-generic objects to a generic type in scenarios where the compiler would otherwise complain.
`OfType<T>`	Remove items that do not match a specified type.
`Distinct`	Remove duplicate items.
`Except`, `Intersect`, `Union`	Perform operations that return sets. Sets cannot have duplicate items. Although the inputs can be any sequence and so the inputs can have duplicates, the result is always a set.
`DistinctBy`, `ExceptBy`, `IntersectBy`, `UnionBy`,	Allow the comparison to be performed on a subset of the item rather than the entire item. For example, instead of removing duplicates with `Distinct` by comparing an

`MinBy` , `MaxBy`	entire `Person` object, you could remove duplicates with `DistinctBy` by comparing just their `LastName` and `DateOfBirth` .
`Chunk`	Divide a sequence into sized batches.
`Append` , `Concat` , `Prepend`	Perform sequence-combining operations.
`Zip`	Perform a match operation on two or three sequences based on the position of items, for example, the item at position 1 in the first sequence matches the item at position 1 in the second sequence.
`ToArray` , `ToList` , `ToDictionary` , `ToHashSet` , `ToLookup`	Convert the sequence into an array or collection. These are the only extension methods that execute the LINQ expression.

The `Enumerable` class also has some methods that are not extension methods, as shown in the following table:

Method	Description
`Empty<T>`	Returns an empty sequence of the specified type `T` . It is useful for passing an empty sequence to a method that requires an `IEnumerable<T>` .
`Range`	Returns a sequence of integers from the `start` value with `count` items. For example, `Enumerable.Range(start: 5, count: 3)` would contain the integers 5, 6, and 7.
`Repeat`	Returns a sequence that contains the same element repeated `count` times. For example, `Enumerable.Repeat(element: "5", count: 3)` would contain the `string` values `"5"` , `"5"` , and `"5"` .

Understanding deferred execution

LINQ uses **deferred execution**. It is important to understand that calling most of these extension methods does not execute the query and get the results. Most of these extension methods return a LINQ expression that represents a *question*, not an *answer*. Let's explore:

1. Use your preferred code editor to create a new project, as defined in the following list:
   - Project template: **Console App** / `console`
   - Project file and folder: `LinqWithObjects`
   - Workspace/solution file and folder: `Chapter11`
2. In the project file, globally and statically import the `System.Console` class.
3. Add a new class file name `Program.Helpers.cs` .
4. In `Program.Helpers.cs` , define a partial `Program` class with a method to output a section title, as shown in the following code:

```
partial class Program
{
  static void SectionTitle(string title)
  {
    ConsoleColor previousColor = ForegroundColor;
    ForegroundColor = ConsoleColor.DarkYellow;
    WriteLine("*");
    WriteLine($"* {title}");
    WriteLine("*");
    ForegroundColor = previousColor;
  }
}
```

1. In `Program.cs` , delete the existing statements and then add statements to define a sequence of `string` values for people who work in an office, as shown in the following code:

```
// a string array is a sequence that implements IEnumerable<string>
string[] names = new[] { "Michael", "Pam", "Jim", "Dwight",
  "Angela", "Kevin", "Toby", "Creed" };
SectionTitle("Deferred execution");
// Question: Which names end with an M?
// (written using a LINQ extension method)
```

```
var query1 = names.Where(name => name.EndsWith("m"));
// Question: Which names end with an M?
// (written using LINQ query comprehension syntax)
var query2 = from name in names where name.EndsWith("m") select name;
```

1. To get the answer, i.e. execute the query, you must **materialize** it by either calling one of the "To" methods like `ToArray` or `ToLookup` or by enumerating the query, as shown in the following code:

```
// Answer returned as an array of strings containing Pam and Jim
string[] result1 = query1.ToArray();
// Answer returned as a list of strings containing Pam and Jim
List<string> result2 = query2.ToList();
// Answer returned as we enumerate over the results
foreach (string name in query1)
{
  WriteLine(name); // outputs Pam
  names[2] = "Jimmy"; // change Jim to Jimmy
  // on the second iteration Jimmy does not end with an M
}
```

1. Run the console app and note the result, as shown in the following output:

```
Pam
```

Due to deferred execution, after outputting the first result, `Pam`, if the original array values change, then by the time we loop back around, there are no more matches because `Jim` has become `Jimmy` and does not end with an `M`, so only `Pam` is outputted.

Before we get too deep into the weeds, let's slow down and look at some common LINQ extension methods and how to use them, one at a time.

Filtering entities with Where

The most common reason for using LINQ is to filter items in a sequence using the `Where` extension method. Let's explore filtering by defining a sequence of names and then applying LINQ operations to it:

1. In the project file, add an element to remove the `System.Linq` namespace from automatically being imported globally, as shown highlighted in the following markup:

```
<ItemGroup>
    <Using Include="System.Console" Static="true" />
    <Using Remove="System.Linq" />
</ItemGroup>
```

1. In `Program.cs`, attempt to call the `Where` extension method on the array of names, as shown in the following code:

```
SectionTitle("Writing queries");
var query = names.W
```

1. As you try to type the `Where` method, note that it is missing from the IntelliSense list of members of a `string` array, as shown in *Figure 11.1*:

Figure 11.1: IntelliSense with the Where extension method missing

This is because `Where` is an extension method. It does not exist on the array type. To make the `Where` extension method available, we must import the `System.Linq` namespace. This is implicitly imported by default in new .NET 6 projects, but we removed it.

1. In the project file, comment out the element that removed `System.Linq`, as shown in the following code:

```
<!--<Using Remove="System.Linq" />-->
```

1. Retype the `Where` method and note that the IntelliSense list now includes the extension methods added by the `Enumerable` class, as shown in *Figure 11.2*:

Figure 11.2: IntelliSense showing LINQ Enumerable extension methods now

2. As you type the parentheses for the `Where` method, IntelliSense tells us that to call `Where`, we must pass in an instance of a `Func<string, bool>` delegate.
3. Enter an expression to create a new instance of a `Func<string, bool>` delegate, and for now note that we have not yet supplied a method name because we will define it in the next step, as shown in the following code:

```
var query = names.Where(new Func<string, bool>( ))
```

The `Func<string, bool>` delegate tells us that for each `string` variable passed to the method, the method must return a `bool` value. If the method returns `true`, it indicates that we should include the `string` in the results, and if the method returns `false`, it indicates that we should exclude it.

Targeting a named method

Let's define a method that only includes names that are longer than four characters:

1. Add a new class named `Program.Functions.cs`.
2. In `Program.Functions.cs`, define a partial `Program` class with a method that will include only names longer than four characters, as shown in the following code:

```
partial class Program
{
  static bool NameLongerThanFour(string name)
  {
    return name.Length > 4;
  }
}
```

1. In `Program.cs`, pass the method's name into the `Func<string, bool>` delegate, and then loop through the query items, as shown highlighted in the following code:

```
var query = names.Where(
  new Func<string, bool>(NameLongerThanFour));
foreach (string item in query)
{
  WriteLine(item);
}
```

1. Run the code and view the results, noting that only names longer than four letters are listed, as shown in the following output:

```
Michael
Dwight
Angela
Kevin
Creed
```

Simplifying the code by removing the explicit delegate instantiation

We can simplify the code by deleting the explicit instantiation of the `Func<string, bool>` delegate because the C# compiler can instantiate the delegate for us:

1. To help you learn by seeing progressively improved code, copy and paste the query
2. Comment out the first example, as shown in the following code:

```
// var query = names.Where(
//   new Func<string, bool>(NameLongerThanFour));
```

1. Modify the copy to remove the explicit instantiation of the delegate, as shown in the following code:

```
var query = names.Where(NameLongerThanFour);
```

1. Run the code and note that it has the same behavior.

Targeting a lambda expression

We can simplify our code even further using a **lambda expression** in place of a named method.

Although it can look complicated at first, a lambda expression is simply a *nameless function*. It uses the `=>` (read as "goes to") symbol to indicate the return value:

1. Copy and paste the query, comment the second example, and modify the query, as shown in the following code:

```
var query = names.Where(name => name.Length > 4);
```

Note that the syntax for a lambda expression includes all the important parts of the `NameLongerThanFour` method, but nothing more. A lambda expression only needs to define the following:

- The names of input parameters: `name`
- A return value expression: `name.Length > 4`

The type of the `name` input parameter is inferred from the fact that the sequence contains `string` values, and the return type must be a `bool` value as defined by the delegate for `Where` to work, so the expression after the `=>` symbol must return a `bool` value.

The compiler does most of the work for us, so our code can be as concise as possible.

1. Run the code and note that it has the same behavior.

Sorting entities

Other commonly used extension methods are `OrderBy` and `ThenBy`, used for sorting a sequence.

Extension methods can be chained if the previous method returns another sequence, that is, a type that implements the `IEnumerable<T>` interface.

Sorting by a single property using OrderBy

Let's continue working with the current project to explore sorting:

1. Append a call to `OrderBy` to the end of the existing query, as shown in the following code:

```
var query = names
  .Where(name => name.Length > 4)
  .OrderBy(name => name.Length);
```

Good Practice: Format the LINQ statement so that each extension method call happens on its own line to make them easier to read.

1. Run the code and note that the names are now sorted by shortest first, as shown in the following output:

```
Kevin
Creed
Dwight
Angela
Michael
```

To put the longest name first, you would use `OrderByDescending`.

Sorting by a subsequent property using ThenBy

We might want to sort by more than one property, for example, to sort names of the same length in alphabetical order:

1. Add a call to the `ThenBy` method at the end of the existing query, as shown highlighted in the following code:

```
var query = names
  .Where(name => name.Length > 4)
  .OrderBy(name => name.Length)
  .ThenBy(name => name);
```

1. Run the code and note the slight difference in the following sort order. Within a group of names of the same length, the names are sorted alphabetically by the full value of the `string`, so `Creed` comes before `Kevin`, and `Angela` comes before `Dwight`, as shown in the following output:

```
Creed
Kevin
Angela
```

```
Dwight
Michael
```

Declaring a query using var or a specified type

While writing a LINQ expression it is convenient to use `var` to declare the query object. This is because the type frequently changes as you work on the LINQ expression. For example, our query started as an `IEnumerable<string>` and is currently an `IOrderedEnumerable<string>`:

1. Hover your mouse over the `var` keyword and note that its type is `IOrderedEnumerable<string>`
2. Replace `var` with the actual type, as shown highlighted in the following code:

```
IOrderedEnumerable<string> query = names
  .Where(name => name.Length > 4)
  .OrderBy(name => name.Length)
  .ThenBy(name => name);
```

Good Practice: Once you have finished working on a query, you could change the declared type from `var` to the actual type to make it clearer what the type is. This is easy because your code editor can tell you what it is.

Filtering by type

The `Where` extension method is great for filtering by values, such as text and numbers. But what if the sequence contains multiple types, and you want to filter by a specific type and respect any inheritance hierarchy?

Imagine that you have a sequence of exceptions. There are hundreds of exception types that form a complex hierarchy, as partially shown in *Figure 11.3*:

Figure 11.3: A partial exception inheritance hierarchy

Let's explore filtering by type:

1. In `Program.cs`, define a list of exception-derived objects, as shown in the following code:

```
SectionTitle("Filtering by type");
List<Exception> exceptions = new()
{
  new ArgumentException(),
  new SystemException(),
  new IndexOutOfRangeException(),
  new InvalidOperationException(),
  new NullReferenceException(),
  new InvalidCastException(),
  new OverflowException(),
  new DivideByZeroException(),
  new ApplicationException()
};
```

1. Write statements using the `OfType<T>` extension method to remove exceptions that are not arithmetic exceptions and write only the arithmetic exceptions to the console, as shown in the following code:

```
IEnumerable<ArithmeticException> arithmeticExceptionsQuery =
  exceptions.OfType<ArithmeticException>();
foreach (ArithmeticException exception in arithmeticExceptionsQuery)
{
  WriteLine(exception);
}
```

1. Run the code and note that the results only include exceptions of the `ArithmeticException` type, or the `ArithmeticException`-derived types, as shown in the following output:

```
System.OverflowException: Arithmetic operation resulted in an overflow.
System.DivideByZeroException: Attempted to divide by zero.
```

Working with sets and bags using LINQ

Sets are one of the most fundamental concepts in mathematics. A **set** is a collection of one or more unique objects. A **multiset**, aka **bag**, is a collection of one or more objects that can have duplicates.

You might remember being taught about Venn diagrams in school. Common set operations include the **intersect** or **union** between sets.

Let's write some code that will define three arrays of `string` values for cohorts of apprentices and then perform some common set and multiset operations on them:

1. In `Program.Helpers.cs`, add the following method that outputs any sequence of `string` variables as a comma-separated single `string` to the console output, along with an optional description, as shown in the following code:

```
static void Output(IEnumerable<string> cohort, string description = "")
{
  if (!string.IsNullOrEmpty(description))
  {
    WriteLine(description);
  }
  Write(" ");
  WriteLine(string.Join(", ", cohort.ToArray()));
  WriteLine();
}
```

1. In `Program.cs`, add statements to define three arrays of names, output them, and then perform various set operations on them, as shown in the following code:

```
string[] cohort1 = new[]
  { "Rachel", "Gareth", "Jonathan", "George" };
string[] cohort2 = new[]
  { "Jack", "Stephen", "Daniel", "Jack", "Jared" };
string[] cohort3 = new[]
  { "Declan", "Jack", "Jack", "Jasmine", "Conor" };
SectionTitle("The cohorts");
Output(cohort1, "Cohort 1");
Output(cohort2, "Cohort 2");
Output(cohort3, "Cohort 3");
SectionTitle("Set operations");
Output(cohort2.Distinct(), "cohort2.Distinct()");
Output(cohort2.DistinctBy(name => name.Substring(0, 2)),
  "cohort2.DistinctBy(name => name.Substring(0, 2)):");
Output(cohort2.Union(cohort3), "cohort2.Union(cohort3)");
Output(cohort2.Concat(cohort3), "cohort2.Concat(cohort3)");
Output(cohort2.Intersect(cohort3), "cohort2.Intersect(cohort3)");
Output(cohort2.Except(cohort3), "cohort2.Except(cohort3)");
Output(cohort1.Zip(cohort2,(c1, c2) => $"{c1} matched with {c2}"),
  "cohort1.Zip(cohort2)");
```

1. Run the code and view the results, as shown in the following output:

```
Cohort 1
  Rachel, Gareth, Jonathan, George
Cohort 2
  Jack, Stephen, Daniel, Jack, Jared
Cohort 3
  Declan, Jack, Jack, Jasmine, Conor
cohort2.Distinct()
  Jack, Stephen, Daniel, Jared
cohort2.DistinctBy(name => name.Substring(0, 2)):
  Jack, Stephen, Daniel
cohort2.Union(cohort3)
  Jack, Stephen, Daniel, Jared, Declan, Jasmine, Conor
cohort2.Concat(cohort3)
  Jack, Stephen, Daniel, Jack, Jared, Declan, Jack, Jack, Jasmine, Conor
cohort2.Intersect(cohort3)
  Jack
cohort2.Except(cohort3)
  Stephen, Daniel, Jared
cohort1.Zip(cohort2)
  Rachel matched with Jack, Gareth matched with Stephen, Jonathan matched with Daniel, George ma
```

With `Zip`, if there are unequal numbers of items in the two sequences, then some items will not have a matching partner. Those without a partner, like `Jared`, will not be included in the result.

For the `DistinctBy` example, instead of removing duplicates by comparing the whole name, we define a lambda key selector to remove duplicates by comparing the first two characters, so `Jared` is removed because `Jack` already is a name that starts with `Ja`.

So far, we have used the LINQ to Objects provider to work with in-memory objects. Next, we will use the LINQ to Entities provider to work with entities stored in a database.

Using LINQ with EF Core

We have looked at LINQ queries that filter and sort, but none that change the shape of the items in the sequence. This is called **projection** because it's about projecting items of one shape into another shape. To learn about projection, it is best to have some more complex types to work with, so in the next project, instead of using `string` sequences, we will use sequences of entities from the Northwind sample database.

I will give instructions to use SQLite because it is cross-platform but if you prefer to use SQL Server then feel free to do so. I have included some commented code to enable SQL Server if you choose.

Building an EF Core model

We must define an EF Core model to represent the database and tables that we will work with. We will define the model manually to take complete control and to prevent a relationship from being automatically defined between the `Categories` and `Products` tables. Later, you will use LINQ to join the two entity sets:

1. Use your preferred code editor to add a new **Console App** / `console` project named `LinqWithEFCore` to the `Chapter11` solution/workspace.
2. In Visual Studio Code, select `LinqWithEFCore` as the active OmniSharp project.
3. In the `LinqWithEFCore` project, add a package reference to the EF Core provider for SQLite and/or SQL Server, as shown in the following markup:

```
<ItemGroup>
  <PackageReference
    Include="Microsoft.EntityFrameworkCore.Sqlite"
    Version="7.0.0" />
  <PackageReference
    Include="Microsoft.EntityFrameworkCore.SqlServer"
```

```
    Version="7.0.0" />
</ItemGroup>
```

Early Access Program: Until the final release of .NET 7, the version numbers of packages will be for previews and release candidates. For example, `7.0.0-preview.1.22109.13`. You can change the version to `7.0.0-*` to automatically upgrade to the latest packages as monthly previews and release candidates are released.

1. Build the project to restore packages.
2. Copy the `Northwind4Sqlite.sql` file into the `LinqWithEFCore` folder.
3. At a command prompt or terminal, create the Northwind database by executing the following command:

```
sqlite3 Northwind.db -init Northwind4Sqlite.sql
```

1. Be patient because this command might take a while to create the database structure. Eventually you will see the SQLite command prompt, as shown in the following output:

```
 -- Loading resources from Northwind.sql
SQLite version 3.36.0 2021-08-02 15:20:15
Enter ".help" for usage hints.
sqlite>
```

1. To exit SQLite command mode, press Ctrl + C on Windows or cmd + D on macOS.
2. Add three class files to the project, named `Northwind.cs`, `Category.cs`, and `Product.cs`.
3. Modify the class file named `Northwind.cs`, as shown in the following code:

```
using Microsoft.EntityFrameworkCore; // DbContext, DbSet<T>
namespace Packt.Shared;
public class Northwind : DbContext
{
  public DbSet<Category> Categories { get; set; } = null!;
  public DbSet<Product> Products { get; set; } = null!;
  protected override void OnConfiguring(
    DbContextOptionsBuilder optionsBuilder)
  {
    string path = Path.Combine(
      Environment.CurrentDirectory, "Northwind.db");
    optionsBuilder.UseSqlite($"Filename={path}");
    /*
    string connection = "Data Source=.;" +
        "Initial Catalog=Northwind;" +
        "Integrated Security=true;" +
        "MultipleActiveResultSets=true;";
    optionsBuilder.UseSqlServer(connection);
    */
  }
  protected override void OnModelCreating(
    ModelBuilder modelBuilder)
  {
    if (Database.ProviderName.Contains("Sqlite"))
    {
      modelBuilder.Entity<Product>()
        .Property(product => product.UnitPrice)
        .HasConversion<double>();
    }
  }
}
```

1. Modify the class file named `Category.cs`, as shown in the following code:

```
using System.ComponentModel.DataAnnotations;
namespace Packt.Shared;
public class Category
{
  public int CategoryId { get; set; }
  [Required]
  [StringLength(15)]
```

```
  public string CategoryName { get; set; } = null!;
  public string? Description { get; set; }
}
```

1. Modify the class file named `Product.cs`, as shown in the following code:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace Packt.Shared;
public class Product
{
  public int ProductId { get; set; }
  [Required]
  [StringLength(40)]
  public string ProductName { get; set; } = null!;
  public int? SupplierId { get; set; }
  public int? CategoryId { get; set; }
  [StringLength(20)]
  public string? QuantityPerUnit { get; set; }
  [Column(TypeName = "money")] // required for SQL Server provider
  public decimal? UnitPrice { get; set; }
  public short? UnitsInStock { get; set; }
  public short? UnitsOnOrder { get; set; }
  public short? ReorderLevel { get; set; }
  public bool Discontinued { get; set; }
}
```

1. Build the project and fix any compiler errors.

   If you are using Visual Studio for Windows, then the compiled application executes in the `LinqWithEFCore\bin\Debug\net7.0` folder so it will not find the database file unless we indicate that it should always be copied to the output directory.

2. In **Solution Explorer**, right-click the `Northwind.db` file and select **Properties**.
3. In **Properties**, set **Copy to Output Directory** to **Copy always**.

Filtering and sorting sequences

Now let's write statements to filter and sort sequences of rows from the tables:

1. In the `LinqWithEFCore` project, add a new class file name `Program.Helpers.cs`.
2. In `Program.Helpers.cs`, define a partial `Program` class with a method to output a section title, as shown in the following code:

```
partial class Program
{
  static void SectionTitle(string title)
  {
    ConsoleColor previousColor = ForegroundColor;
    ForegroundColor = ConsoleColor.DarkYellow;
    WriteLine("*");
    WriteLine($"* {title}");
    WriteLine("*");
    ForegroundColor = previousColor;
  }
}
```

1. In the `LinqWithEFCore` project, add a new class file named `Program.Functions.cs`.
2. In `Program.Functions.cs`, define a partial `Program` class and add a method to filter and sort products, as shown in the following code:

```
using Packt.Shared; // Northwind, Category, Product
using Microsoft.EntityFrameworkCore; // DbSet<T>
partial class Program
{
  static void FilterAndSort()
```

```
  {
    SectionTitle("Filter and sort");
    using (Northwind db = new())
    {
      DbSet<Product> allProducts = db.Products;
      IQueryable<Product> filteredProducts =
        allProducts.Where(product => product.UnitPrice < 10M);
      IOrderedQueryable<Product> sortedAndFilteredProducts =
        filteredProducts.OrderByDescending(product => product.UnitPrice);
      WriteLine("Products that cost less than $10:");
      foreach (Product p in sortedAndFilteredProducts)
      {
        WriteLine("{0}: {1} costs {2:$#,##0.00}",
          p.ProductId, p.ProductName, p.UnitPrice);
      }
      WriteLine();
    }
  }
}
```

`DbSet<T>` implements `IEnumerable<T>`, so LINQ can be used to query and manipulate collections of entities in models built for EF Core.

(Actually, I should say `TEntity` instead of `T` but the name of this generic type has no functional effect. The only requirement is that the type is a `class`. The name just indicates the class is expected to be an entity model.)

You might have also noticed that the sequences implement `IQueryable<T>` (or `IOrderedQueryable<T>` after a call to an ordering LINQ method) instead of `IEnumerable<T>` or `IOrderedEnumerable<T>`.

This is an indication that we are using a LINQ provider that builds the query using expression trees. They represent code in a tree-like data structure and enable the creation of dynamic queries, which is useful for building LINQ queries for external data providers like SQLite.

The LINQ expression will be converted into another query language, such as SQL. Enumerating the query with `foreach` or calling a method such as `ToArray` will force the execution of the query and materialize the results.

1. In `Program.cs`, call the `FilterAndSort` method.
2. Run the code and view the result, as shown in the following output:

```
Products that cost less than $10:
41: Jack's New England Clam Chowder costs $9.65
45: Rogede sild costs $9.50
47: Zaanse koeken costs $9.50
19: Teatime Chocolate Biscuits costs $9.20
23: Tunnbröd costs $9.00
75: Rhönbräu Klosterbier costs $7.75
54: Tourtière costs $7.45
52: Filo Mix costs $7.00
13: Konbu costs $6.00
24: Guaraná Fantástica costs $4.50
33: Geitost costs $2.50
```

Although this query outputs the information we want, it does so inefficiently because it gets all columns from the `Products` table instead of just the three columns we need. Let's log the generated SQL:

1. In the `FilterAndSort` method, add a statement to output the SQL, as shown in the following code:

```
WriteLine(sortedAndFilteredProducts.ToQueryString());
```

1. Run the code and view the result, as shown in the following partial output:

```
SELECT "p"."ProductId", "p"."CategoryId", "p"."Discontinued", "p"."ProductName", "p"."QuantityPe
FROM "Products" AS "p"
WHERE "p"."UnitPrice" < 10.0
ORDER BY "p"."UnitPrice" DESC
```

Projecting sequences into new types

Before we look at projection, we need to review object initialization syntax. If you have a class defined, then you can instantiate an object using the class name, `new()` , and curly braces to set initial values for fields and properties, as shown in the following code:

```
// Person.cs
public class Person
{
  public string Name { get; set; }
  public DateTime DateOfBirth { get; set; }
}
// Program.cs
Person knownTypeObject = new()
{
  Name = "Boris Johnson",
  DateOfBirth = new(year: 1964, month: 6, day: 19)
};
```

C# 3.0 and later allow instances of **anonymous types** to be instantiated using the `var` keyword, as shown in the following code:

```
var anonymouslyTypedObject = new
{
  Name = "Boris Johnson",
  DateOfBirth = new DateTime(year: 1964, month: 6, day: 19)
};
```

Although we did not specify a type, the compiler can infer an anonymous type from the setting of two properties named `Name` and `DateOfBirth` . The compiler can infer the types of the two properties from the values assigned: a literal `string` and a new instance of a date/time value.

This capability is especially useful when writing LINQ queries to project an existing type into a new type without having to explicitly define the new type. Since the type is anonymous, this can only work with `var` -declared local variables.

Let's make the SQL command executed against the database table more efficient by adding a call to the `Select` method to project instances of the `Product` class into instances of a new anonymous type with only three properties:

1. In `Program.Functions.cs` , in the `FilterAndSort` method, add a statement to extend the LINQ query to use the `Select` method to return only the three properties (that is, table columns) that we need, and modify the `foreach` statement to use the `var` keyword and the projection LINQ expression, as shown highlighted in the following code:

```
IOrderedQueryable<Product> sortedAndFilteredProducts =
  filteredProducts.OrderByDescending(product => product.UnitPrice);
var projectedProducts = sortedAndFilteredProducts
  .Select(product => new // anonymous type
  {
    product.ProductId,
    product.ProductName,
    product.UnitPrice
  });
WriteLine(projectedProducts.ToQueryString());
WriteLine("Products that cost less than $10:");
foreach (var p in projectedProducts)
{
```

1. Hover your mouse over the `new` keyword in the `Select` method call and the `var` keyword in the `foreach` statement and note that it is an anonymous type, as shown in *Figure 11.4*:

Figure 11.4: An anonymous type used during LINQ projection

2. Run the code and confirm that the output is the same as before and the generated SQL is more efficient, as shown in the following output:

```
SELECT "p"."ProductId", "p"."ProductName", "p"."UnitPrice"
FROM "Products" AS "p"
WHERE "p"."UnitPrice" < 10.0
ORDER BY "p"."UnitPrice" DESC
```

Joining and grouping sequences

There are two extension methods for joining and grouping:

- `Join` : This method has four parameters: the sequence that you want to join with, the property or properties on the *left* sequence to match on, the property or properties on the *right* sequence to match on, and a projection.
- `GroupJoin` : This method has the same parameters, but it combines the matches into a group object with a `Key` property for the matching value and an `IEnumerable<T>` type for the multiple matches.

Joining sequences

Let's explore these methods when working with two tables: `Categories` and `Products` :

1. In `Program.Functions.cs` , add a method to select categories and products, join them, and output them, as shown in the following code:

```
static void JoinCategoriesAndProducts()
{
  SectionTitle("Join categories and products");
  using (Northwind db = new())
  {
    // join every product to its category to return 77 matches
    var queryJoin = db.Categories.Join(
      inner: db.Products,
      outerKeySelector: category => category.CategoryId,
      innerKeySelector: product => product.CategoryId,
      resultSelector: (c, p) =>
        new { c.CategoryName, p.ProductName, p.ProductId });
    foreach (var item in queryJoin)
    {
      WriteLine("{0}: {1} is in {2}.",
        arg0: item.ProductId,
        arg1: item.ProductName,
        arg2: item.CategoryName);
    }
  }
}
```

In a join, there are two sequences, `outer` and `inner`. In the previous example, `categories` is the outer sequence and `products` is the inner sequence.

1. In `Program.cs`, call the `JoinCategoriesAndProducts` method.
2. Run the code and view the results. Note that there is a single line of output for each of the 77 products, as shown in the following output (edited to only include the first four items):

```
1: Chai is in Beverages.
2: Chang is in Beverages.
3: Aniseed Syrup is in Condiments.
4: Chef Anton's Cajun Seasoning is in Condiments.
...
```

1. At the end of the existing query, call the `OrderBy` method to sort by `CategoryName`, as shown in the following code:

```
.OrderBy(cp => cp.CategoryName);
```

1. Run the code and view the results. Note that there is a single line of output for each of the 77 products, and the results show all products in the `Beverages` category first, then the `Condiments` category, and so on, as shown in the following partial output:

```
1: Chai is in Beverages.
2: Chang is in Beverages.
24: Guaraná Fantástica is in Beverages.
34: Sasquatch Ale is in Beverages.
...
```

Group-joining sequences

1. In `Program.Functions.cs`, add a method to group and join, show the group name, and then show all the items within each group, as shown in the following code:

```
static void GroupJoinCategoriesAndProducts()
{
  SectionTitle("Group join categories and products");
  using (Northwind db = new())
  {
    // group all products by their category to return 8 matches
    var queryGroup = db.Categories.AsEnumerable().GroupJoin(
      inner: db.Products,
      outerKeySelector: category => category.CategoryId,
      innerKeySelector: product => product.CategoryId,
      resultSelector: (c, matchingProducts) => new
      {
        c.CategoryName,
        Products = matchingProducts.OrderBy(p => p.ProductName)
      });
    foreach (var category in queryGroup)
    {
      WriteLine("{0} has {1} products.",
        arg0: category.CategoryName,
        arg1: category.Products.Count());
      foreach (var product in category.Products)
      {
        WriteLine($" {product.ProductName}");
      }
    }
  }
}
```

If we had not called the `AsEnumerable` method, then a runtime exception would have been thrown, as shown in the following output:

```
Unhandled exception. System.ArgumentException:  Argument type 'System.Linq.IOrderedQueryable`1[Pa
```

This is because not all LINQ extension methods can be converted from expression trees into some other query syntax like SQL. In these cases, we can convert from `IQueryable<T>` to `IEnumerable<T>` by calling the `AsEnumerable` method, which forces query processing to use LINQ to EF Core only to bring the data into the application and then use LINQ to Objects to execute more complex processing in memory. But, often, this is less efficient.

1. In `Program.cs`, call the `GroupJoinCategoriesAndProducts` method.
2. Run the code, view the results, and note that the products inside each category have been sorted by their name, as defined in the query, and as shown in the following partial output:

```
Beverages has 12 products.
  Chai
  Chang
  ...
Condiments has 12 products.
  Aniseed Syrup
  Chef Anton's Cajun Seasoning
  ...
```

Aggregating sequences

There are LINQ extension methods to perform aggregation functions, such as `Average` and `Sum`. Let's write some code to see some of these methods in action aggregating information from the `Products` table:

1. In `Program.Functions.cs`, add a method to show the use of the aggregation extension methods, as shown in the following code:

```
static void AggregateProducts()
{
  SectionTitle("Aggregate products");
  using (Northwind db = new())
  {
    WriteLine("{0,-25} {1,10}",
      arg0: "Product count:",
      arg1: db.Products.Count());
    WriteLine("{0,-25} {1,10:$#,##0.00}",
      arg0: "Highest product price:",
      arg1: db.Products.Max(p => p.UnitPrice));
    WriteLine("{0,-25} {1,10:N0}",
      arg0: "Sum of units in stock:",
      arg1: db.Products.Sum(p => p.UnitsInStock));
    WriteLine("{0,-25} {1,10:N0}",
      arg0: "Sum of units on order:",
      arg1: db.Products.Sum(p => p.UnitsOnOrder));
    WriteLine("{0,-25} {1,10:$#,##0.00}",
      arg0: "Average unit price:",
      arg1: db.Products.Average(p => p.UnitPrice));
    WriteLine("{0,-25} {1,10:$#,##0.00}",
      arg0: "Value of units in stock:",
      arg1: db.Products
        .Sum(p => p.UnitPrice * p.UnitsInStock));
  }
}
```

1. In `Program.cs`, call the `AggregateProducts` method.
2. Run the code and view the result, as shown in the following output:

```
Product count:                    77
Highest product price:       $263.50
Sum of units in stock:         3,119
Sum of units on order:           780
Average unit price:           $28.87
Value of units in stock:  $74,050.85
```

Sweetening LINQ syntax with syntactic sugar

C# 3.0 introduced some new language keywords in 2008 to make it easier for programmers with experience with SQL to write LINQ queries. This syntactic sugar is sometimes called the **LINQ query comprehension syntax**.

Consider the following array of `string` values:

```
string[] names = new[] { "Michael", "Pam", "Jim", "Dwight",
  "Angela", "Kevin", "Toby", "Creed" };
```

To filter and sort the names, you could use extension methods and lambda expressions, as shown in the following code:

```
var query = names
  .Where(name => name.Length > 4)
  .OrderBy(name => name.Length)
  .ThenBy(name => name);
```

Or you could achieve the same results by using query comprehension syntax, as shown in the following code:

```
var query = from name in names
  where name.Length > 4
  orderby name.Length, name
  select name;
```

The compiler changes the query comprehension syntax to the equivalent extension methods and lambda expressions for you.

> The `select` keyword is always required for LINQ query comprehension syntax. The `Select` extension method is optional when using extension methods and lambda expressions because if you do not call `Select`, then the whole item is implicitly selected.

Not all extension methods have a C# keyword equivalent, for example, the `Skip` and `Take` extension methods, which are commonly used to implement paging for lots of data.

A query that skips and takes cannot be written using only the query comprehension syntax, so we could write the query using all extension methods, as shown in the following code:

```
var query = names
  .Where(name => name.Length > 4)
  .Skip(80)
  .Take(10);
```

Or you can wrap query comprehension syntax in parentheses and then switch to using extension methods, as shown in the following code:

```
var query = (from name in names
  where name.Length > 4
  select name)
  .Skip(80)
  .Take(10);
```

> **Good Practice**: Learn both extension methods with lambda expressions and the query comprehension syntax ways of writing LINQ queries, because you are likely to have to maintain code that uses both.

Using multiple threads with parallel LINQ

By default, only one thread is used to execute a LINQ query. **Parallel LINQ** (**PLINQ**) is an easy way to enable multiple threads to execute a LINQ query.

> **Good Practice**: Do not assume that using parallel threads will improve the performance of your applications. Always measure real-world timings and resource usage.

Creating an app that benefits from multiple threads

To see it in action, we will start with some code that only uses a single thread to calculate Fibonacci numbers for 45 integers. We will use the `StopWatch` type to measure the change in performance.

We will use operating system tools to monitor the CPU and CPU core usage. If you do not have multiple CPUs or at least multiple cores, then this exercise won't show much!

1. Use your preferred code editor to add a new **Console App** / `console` project named `LinqInParallel` to the `Chapter11` solution/workspace.
2. In Visual Studio Code, select `LinqInParallel` as the active OmniSharp project.
3. In `Program.cs`, delete the existing statements and then import the `System.Diagnostics` namespace so that we can use the `StopWatch` type, and add statements to create a stopwatch to record timings, wait for a keypress before starting the timer, create 45 integers, calculate the last Fibonacci number for each of them, stop the timer, and display the elapsed milliseconds, as shown in the following code:

```
using System.Diagnostics; // Stopwatch
Write("Press ENTER to start. "); ReadLine();
Stopwatch watch = Stopwatch.StartNew();
watch.Start();
int max = 45;
IEnumerable<int> numbers = Enumerable.Range(start: 1, count: max);
WriteLine($"Calculating Fibonacci sequence up to term {max}. Please wait...");
int[] fibonacciNumbers = numbers
  .Select(number => Fibonacci(number))
  .ToArray();
watch.Stop();
WriteLine("{0:#,##0} elapsed milliseconds.",
  arg0: watch.ElapsedMilliseconds);
Write("Results:");
foreach (int number in fibonacciNumbers)
{
  Write($" {number:N0}");
}
static int Fibonacci(int term) =>
  term switch
  {
    1 => 0,
    2 => 1,
    _ => Fibonacci(term - 1) + Fibonacci(term - 2)
  };
```

1. Run the console app, but do not press **Enter** to start the stopwatch yet because we need to make sure a monitoring tool is showing processor activity.

Using Windows

If you are using Windows:

1. Right-click on the Windows **Start** button or press Ctrl + Alt + Delete, and then click on **Task Manager**.
2. At the bottom of the **Task Manager** window, click **More details**.
3. At the top of the **Task Manager** window, click on the **Performance** tab.
4. Right-click on the **CPU Utilization** graph, select **Change graph to**, and then select **Logical processors**.

Using macOS

If you are using macOS:

1. Launch **Activity Monitor**.
2. Navigate to **View | Update Frequency Very often (1 sec)**.
3. To see the CPU graphs, navigate to **Window | CPU History**.

For all operating systems

If you are using Windows or macOS or any other OS:

1. Rearrange your monitoring tool and your code editor so that they are side by side.
2. Wait for the CPUs to settle and then press Enter to start the stopwatch and run the query. The result should be a number of elapsed milliseconds, as shown in the following output:

```
Calculating Fibonacci sequence up to term 45. Please wait...
17,624 elapsed milliseconds.
Results: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1,597 2,584 4,181 6,765 10,946 17,711 2
```

The monitoring tool will probably show that one or two CPUs were used the most, alternating over time. Others may execute background tasks at the same time, such as the garbage collector, so the other CPUs or cores won't be completely flat, but the work is certainly not being evenly spread among all the possible CPUs or cores. Also, note that some of the logical processors are maxing out at 100%.

1. In `Program.cs`, modify the query to make a call to the `AsParallel` extension method and to sort the resulting sequence because when processing in parallel the results can become misordered, as shown highlighted in the following code:

```
int[] fibonacciNumbers = numbers.AsParallel()
  .Select(number => Fibonacci(number))
  .OrderBy(number => number)
  .ToArray();
```

Good Practice: Never call `AsParallel` at the end of a query. This does nothing. You must perform at least one operation after the call to `AsParallel` for that operation to be parallelized. .NET 6 introduced a code analyzer that will warn about this type of misuse.

1. Run the code, wait for CPU charts in your monitoring tool to settle, and then press Enter to start the stopwatch and run the query. This time, the application should complete in less time (although it might not be as less as you might hope for—managing those multiple threads takes extra effort!):

```
Calculating Fibonacci sequence up to term 45. Please wait...
9,028 elapsed milliseconds.
Results: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1,597 2,584 4,181 6,765 10,946 17,711 2
```

1. The monitoring tool should show that all CPUs were used equally to execute the LINQ query and note that none of the logical processors max out at 100% because the work is more evenly spread.

Creating your own LINQ extension methods

In *Chapter 6*, *Implementing Interfaces and Inheriting Classes*, you learned how to create your own extension methods. To create LINQ extension methods, all you must do is extend the `IEnumerable<T>` type.

Good Practice: Put your own extension methods in a separate class library so that they can be easily deployed as their own assembly or NuGet package.

We will improve the `Average` extension method as an example. A well-educated school child will tell you that *average* can mean one of three things:

- **Mean**: Sum the numbers and divide by the count.
- **Mode**: The most common number.
- **Median**: The number in the middle of the numbers when ordered.

Microsoft's implementation of the `Average` extension method calculates the *mean*. We might want to define our own extension methods for `Mode` and `Median`:

1. In the `LinqWithEFCore` project, add a new class file named `MyLinqExtensions.cs`.
2. Modify the class, as shown in the following code:

```
namespace System.Linq; // extend Microsoft's namespace
public static class MyLinqExtensions
{
  // this is a chainable LINQ extension method
  public static IEnumerable<T> ProcessSequence<T>(
    this IEnumerable<T> sequence)
  {
    // you could do some processing here
    return sequence;
  }
  public static IQueryable<T> ProcessSequence<T>(
    this IQueryable<T> sequence)
  {
    // you could do some processing here
    return sequence;
  }
  // these are scalar LINQ extension methods
  public static int? Median(
    this IEnumerable<int?> sequence)
  {
    var ordered = sequence.OrderBy(item => item);
    int middlePosition = ordered.Count() / 2;
    return ordered.ElementAt(middlePosition);
  }
  public static int? Median<T>(
    this IEnumerable<T> sequence, Func<T, int?> selector)
  {
    return sequence.Select(selector).Median();
  }
  public static decimal? Median(
    this IEnumerable<decimal?> sequence)
  {
    var ordered = sequence.OrderBy(item => item);
    int middlePosition = ordered.Count() / 2;
    return ordered.ElementAt(middlePosition);
  }
  public static decimal? Median<T>(
    this IEnumerable<T> sequence, Func<T, decimal?> selector)
  {
    return sequence.Select(selector).Median();
  }
  public static int? Mode(
    this IEnumerable<int?> sequence)
  {
    var grouped = sequence.GroupBy(item => item);
    var orderedGroups = grouped.OrderByDescending(
      group => group.Count());
    return orderedGroups.FirstOrDefault()?.Key;
  }
  public static int? Mode<T>(
    this IEnumerable<T> sequence, Func<T, int?> selector)
  {
    return sequence.Select(selector)?.Mode();
  }
  public static decimal? Mode(
    this IEnumerable<decimal?> sequence)
  {
    var grouped = sequence.GroupBy(item => item);
    var orderedGroups = grouped.OrderByDescending(
      group => group.Count());
    return orderedGroups.FirstOrDefault()?.Key;
  }
  public static decimal? Mode<T>(
    this IEnumerable<T> sequence, Func<T, decimal?> selector)
  {
    return sequence.Select(selector).Mode();
  }
}
```

If this class was in a separate class library, to use your LINQ extension methods, you simply need to reference the class library assembly because the `System.Linq` namespace is already implicitly imported.

> **Warning!** All but one of the above extension methods cannot be used with `IQueryable` sequences like those used by LINQ to SQLite or LINQ to SQL Server because we have not implemented a way to translate our code into the underlying query language like SQL.

Trying the chainable extension method

First, we will try chaining the `ProcessSequence` method with other extension methods:

1. In `Program.Functions.cs`, in the `FilterAndSort` method, modify the LINQ query for `Products` to call your custom chainable extension method, as shown highlighted in the following code:

```
DbSet<Product> allProducts = db.Products;
IQueryable<Product> processedProducts = allProducts.ProcessSequence();
IQueryable<Product> filteredProducts = processedProducts
  .Where(product => product.UnitPrice < 10M);
```

1. In `Program.cs`, uncomment the `FilterAndSort` method and comment out any calls to other methods.
2. Run the code and note that you see the same output as before because your method doesn't modify the sequence. But you now know how to extend a LINQ expression with your own functionality.

Trying the mode and median methods

Second, we will try using the `Mode` and `Median` methods to calculate other kinds of average:

1. In `Program.Functions.cs`, add a method to output the mean, median, and mode, for `UnitsInStock` and `UnitPrice` for products, using your custom extension methods and the built-in `Average` extension method, as shown in the following code:

```
static void CustomExtensionMethods()
{
  SectionTitle("Custom aggregate extension methods");
  using (Northwind db = new())
  {
    WriteLine("{0,-25} {1,10:N0}",
      "Mean units in stock:",
      db.Products.Average(p => p.UnitsInStock));
    WriteLine("{0,-25} {1,10:$#,##0.00}",
      "Mean unit price:",
      db.Products.Average(p => p.UnitPrice));

    WriteLine("{0,-25} {1,10:N0}",
      "Median units in stock:",
      db.Products.Median(p => p.UnitsInStock));

    WriteLine("{0,-25} {1,10:$#,##0.00}",
      "Median unit price:",
      db.Products.Median(p => p.UnitPrice));

    WriteLine("{0,-25} {1,10:N0}",
      "Mode units in stock:",
      db.Products.Mode(p => p.UnitsInStock));

    WriteLine("{0,-25} {1,10:$#,##0.00}",
      "Mode unit price:",
      db.Products.Mode(p => p.UnitPrice));
  }
}
```

1. In `Program.cs`, call the `CustomExtensionMethods` method.
2. Run the code and view the result, as shown in the following output:

```
Mean units in stock:              41
Mean unit price:             $28.87
Median units in stock:            26
Median unit price:           $19.50
Mode units in stock:               0
Mode unit price:             $18.00
```

There are four products with a unit price of $18.00. There are five products with 0 units in stock.

Working with LINQ to XML

LINQ to XML is a LINQ provider that allows you to query and manipulate XML.

Generating XML using LINQ to XML

Let's create a method to convert the `Products` table into XML:

1. In the `LinqWithEFCore` project, in `Program.Functions.cs`, import the `System.Xml.Linq` namespace.
2. In `Program.Functions.cs`, add a method to output the products in XML format, as shown in the following code:

```
static void OutputProductsAsXml()
{
  SectionTitle("Output products as XML");
  using (Northwind db = new())
  {
    Product[] productsArray = db.Products.ToArray();
    XElement xml = new("products",
      from p in productsArray
      select new XElement("product",
        new XAttribute("id",  p.ProductId),
        new XAttribute("price", p.UnitPrice),
       new XElement("name", p.ProductName)));
    WriteLine(xml.ToString());
  }
}
```

1. In `Program.cs`, call the `OutputProductsAsXml` method.
2. Run the code, view the result, and note that the structure of the XML generated matches the elements and attributes that the LINQ to XML statement declaratively described in the preceding code, as shown in the following partial output:

```
<products>
  <product id="1" price="18">
    <name>Chai</name>
  </product>
  <product id="2" price="19">
    <name>Chang</name>
  </product>
...
```

Reading XML using LINQ to XML

You might want to use LINQ to XML to easily query or process XML files:

1. In the `LinqWithEFCore` project, add a file named `settings.xml`.
2. Modify its contents, as shown in the following markup:

```
<?xml version="1.0" encoding="utf-8" ?>
<appSettings>
  <add key="color" value="red" />
  <add key="size" value="large" />
  <add key="price" value="23.99" />
</appSettings>
```

If you are using Visual Studio 2022 for Windows, then the compiled application executes in the `LinqWithEFCore\bin\Debug\net7.0` folder so it will not find the `settings.xml` file unless we indicate that it should always be copied to the output directory.

1. In **Solution Explorer**, right-click the `settings.xml` file and select **Properties**.
2. In **Properties**, set **Copy to Output Directory** to **Copy always**.
3. In `Program.Functions.cs`, add a method to complete these tasks, as shown in the following code:
   - Load the XML file.
   - Use LINQ to XML to search for an element named `appSettings` and its descendants named `add`.
   - Project the XML into an array of an anonymous type with `Key` and `Value` properties.
   - Enumerate through the array to show the results:

```
static void ProcessSettings()
{
  string path = Path.Combine(
    Environment.CurrentDirectory, "settings.xml");
  WriteLine($"Settings file path: {path}");
  XDocument doc = XDocument.Load(path);
  var appSettings = doc.Descendants("appSettings")
    .Descendants("add")
    .Select(node => new
    {
      Key = node.Attribute("key")?.Value,
      Value = node.Attribute("value")?.Value
    }).ToArray();
  foreach (var item in appSettings)
  {
    WriteLine($"{item.Key}: {item.Value}");
  }
}
```

1. In `Program.cs`, call the `ProcessSettings` method.
2. Run the console app and view the result, as shown in the following output:

```
Settings file path: C:\cs11dotnet7\Chapter11\LinqWithEFCore\bin\Debug\net7.0\settings.xml
color: red
size: large
price: 23.99
```

Practicing and exploring

Test your knowledge and understanding by answering some questions, get some hands-on practice, and explore with deeper research into the topics covered in this chapter.

Exercise 11.1 – Test your knowledge

Answer the following questions:

1. What are the two required parts of LINQ?
2. Which LINQ extension method would you use to return a subset of properties from a type?
3. Which LINQ extension method would you use to filter a sequence?
4. List five LINQ extension methods that perform aggregation.
5. What is the difference between the `Select` and `SelectMany` extension methods?
6. What is the difference between `IEnumerable<T>` and `IQueryable<T>`? And how do you switch between them?
7. What does the last type parameter `T` in generic `Func` delegates like `Func<T1, T2, T>` represent?
8. What is the benefit of a LINQ extension method that ends with `OrDefault`?
9. Why is query comprehension syntax optional?
10. How can you create your own LINQ extension methods?

Exercise 11.2 – Practice querying with LINQ

In the `Chapter11` solution/workspace, create a console application, named `Ch11Ex02LinqQueries`, that prompts the user for a city and then lists the company names for Northwind customers in that city, as shown in the following output:

```
Enter the name of a city: London
There are 6 customers in London:
  Around the Horn
  B's Beverages
  Consolidated Holdings
  Eastern Connection
  North/South
  Seven Seas Imports
```

Then, enhance the application by displaying a list of all unique cities that customers already reside in as a prompt to the user before they enter their preferred city, as shown in the following output:

```
Aachen, Albuquerque, Anchorage, Århus, Barcelona, Barquisimeto, Bergamo, Berlin, Bern, Boise, Bra
```

Exercise 11.3 – Explore topics

Use the links on the following page to learn more details about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-11---querying-and-manipulating-data-using-linq

Summary

In this chapter, you learned how to write LINQ queries to select, project, filter, sort, join, and group data in many different formats, including XML, which are tasks you will perform every day.

In the next chapter, you will be introduced to web development using ASP.NET Core. In the remaining chapters, you will learn how to implement its major components like Razor Pages, MVC, Web API and Blazor.

12 Introducing Web Development Using ASP.NET Core

Join our book community on Discord

https://packt.link/EarlyAccess

The third and final part of this book is about web development using ASP.NET Core. You will learn how to build cross-platform projects such as websites, web services, and web browser apps.

Microsoft calls platforms for building applications **app models** or **workloads**.

I recommend that you work through this and subsequent chapters sequentially because later chapters will reference projects in earlier chapters, and you will build up sufficient knowledge and skills to tackle the trickier problems in later chapters.

In this chapter, we will cover the following topics:

- Understanding ASP.NET Core
- New features in ASP.NET Core
- Structuring projects
- Building an entity data model for Northwind
- Understanding web development

Understanding ASP.NET Core

Since this book is about C# and .NET, we will learn about app models that use them to build the practical applications that we will encounter in the remaining chapters of this book.

> **Learn More**: Microsoft has extensive guidance for implementing app models in its .NET Application Architecture Guidance documentation, which you can read at the following link: https://www.microsoft.com/net/learn/architecture

Microsoft ASP.NET Core is part of a history of Microsoft technologies used to build websites and services that have evolved over the years:

- **Active Server Pages** (**ASP**) was released in 1996 and was Microsoft's first attempt at a platform for dynamic server-side execution of website code. ASP files contain a mix of HTML and code that executes on the server written in the VBScript language.
- **ASP.NET Web Forms** was released in 2002 with the .NET Framework and was designed to enable non-web developers, such as those familiar with Visual Basic, to quickly create websites by dragging and dropping visual components and writing event-driven code in Visual Basic or C#. Web Forms should be avoided for new .NET Framework web projects in favor of ASP.NET MVC.
- **Windows Communication Foundation** (**WCF**) was released in 2006 and enables developers to build SOAP and REST services. SOAP is powerful but complex, so it should be avoided unless you need advanced features, such as distributed transactions and complex messaging topologies.
- **ASP.NET MVC** was released in 2009 to cleanly separate the concerns of web developers between the **models**, which temporarily store the data; the **views**, which present the data using various formats in the UI;

and the **controllers**, which fetch the model and pass it to a view. This separation enables improved reuse and unit testing.
- **ASP.NET Web API** was released in 2012 and enables developers to create HTTP services (aka REST services) that are simpler and more scalable than SOAP services.
- **ASP.NET SignalR** was released in 2013 and enables real-time communication in websites by abstracting underlying technologies and techniques, such as WebSockets and Long Polling. This enables website features such as live chat or updates to time-sensitive data such as stock prices across a wide variety of web browsers, even when they do not support an underlying technology such as WebSockets.
- **ASP.NET Core** was released in 2016 and combines modern implementations of .NET Framework technologies such as MVC, Web API, and SignalR, with newer technologies such as Razor Pages, gRPC, and Blazor, all running on modern .NET. Therefore, it can execute cross-platform. ASP.NET Core has many project templates to get you started with its supported technologies.

Good Practice: Choose ASP.NET Core to develop websites and web services because it includes web-related technologies that are modern and cross-platform.

Classic ASP.NET versus modern ASP.NET Core

Until modern .NET, ASP.NET was built on top of a large assembly in the .NET Framework named `System.Web.dll` and it was tightly coupled to Microsoft's Windows-only web server named **Internet Information Services** (**IIS**). Over the years, this assembly has accumulated a lot of features, many of which are not suitable for modern cross-platform development.

ASP.NET Core is a major redesign of ASP.NET. It removes the dependency on the `System.Web.dll` assembly and IIS and is composed of modular lightweight packages, just like the rest of modern .NET. Using IIS as the web server is still supported by ASP.NET Core but there is a better option.

You can develop and run ASP.NET Core applications cross-platform on Windows, macOS, and Linux. Microsoft has even created a cross-platform, super-performant web server named **Kestrel**, and the entire stack is open source.

ASP.NET Core 2.2 or later projects default to the new in-process hosting model. This gives a 400% performance improvement when hosting in Microsoft IIS, but Microsoft still recommends using Kestrel for even better performance.

Building websites using ASP.NET Core

Websites are made up of multiple web pages loaded statically from the filesystem or generated dynamically by a server-side technology such as ASP.NET Core. A web browser makes `GET` requests using **Unique Resource Locators** (**URLs**) that identify each page and can manipulate data stored on the server using `POST`, `PUT`, and `DELETE` requests.

With many websites, the web browser is treated as a presentation layer, with almost all the processing performed on the server side. Some JavaScript might be used on the client side to implement some presentation features, such as carousels.

ASP.NET Core provides multiple technologies for building websites:

- **ASP.NET Core Razor Pages** and **Razor class libraries** are ways to dynamically generate HTML for simple websites. You will learn about them in detail in *Chapter 14*, *Building Websites Using ASP.NET Core Razor Pages*.
- **ASP.NET Core MVC** is an implementation of the **Model-View-Controller** (**MVC**) design pattern that is popular for developing complex websites. You will learn about it in detail in *Chapter 15*, *Building Websites Using the Model-View-Controller Pattern*.
- **Blazor** lets you build user interface components using C# and .NET instead of a JavaScript-based UI framework like Angular, React, and Vue. **Blazor WebAssembly** runs your code in the browser like a JavaScript-based framework would. **Blazor Server** runs your code on the server and updates the web page

dynamically. You will learn about Blazor in detail in *Chapter 17*, *Building User Interfaces Using Blazor*. Blazor is not just for building websites; it can also be used to create hybrid mobile and desktop apps.

Building websites using a content management system

Most websites have a lot of content, and if developers had to be involved every time some content needed to be changed, that would not scale well. A **Content Management System** (**CMS**) enables developers to define content structure and templates to provide consistency and good design while making it easy for a non-technical content owner to manage the actual content. They can create new pages or blocks of content, and update existing content, knowing it will look great for visitors with minimal effort.

There is a multitude of CMSs available for all web platforms, like WordPress for PHP or Django CMS for Python. CMSs that support modern .NET include Optimizely Content Cloud, Piranha CMS, and Orchard Core.

The key benefit of using a CMS is that it provides a friendly content management user interface. Content owners log in to the website and manage the content themselves. The content is then rendered and returned to visitors using ASP.NET Core MVC controllers and views, or via web service endpoints, known as a **headless CMS**, to provide that content to "heads" implemented as mobile or desktop apps, in-store touchpoints, or clients built with JavaScript frameworks or Blazor.

This book does not cover .NET CMSs, so I have included links where you can learn more about them in the GitHub repository:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#net-content-management-systems

Building web applications using SPA frameworks

Web applications, also known as **Single-Page Applications** (**SPAs**), are made up of a single web page built with a frontend technology such as Blazor WebAssembly, Angular, React, Vue, or a proprietary JavaScript library that can make requests to a backend web service for getting more data when needed and posting updated data using common serialization formats such as XML and JSON. The canonical examples are Google web apps like Gmail, Maps, and Docs.

With a web application, the client side uses JavaScript frameworks or Blazor WebAssembly to implement sophisticated user interactions, but most of the important processing and data access still happens on the server side, because the web browser has limited access to local system resources.

JavaScript is loosely typed and is not designed for complex projects, so most JavaScript libraries these days use Microsoft TypeScript, which adds strong typing to JavaScript and is designed with many modern language features for handling complex implementations.

.NET SDK has project templates for JavaScript and TypeScript-based SPAs, but we will not spend any time learning how to build JavaScript- and TypeScript-based SPAs in this book, even though these are commonly used with ASP.NET Core as the backend, because this book is about C#, it is not about other languages.

In summary, C# and .NET can be used on both the server side and the client side to build websites, as shown in *Figure 12.1*:

Figure 12.1: The use of C# and .NET to build websites on both the server side and the client side

Building web and other services

Although we will not learn about JavaScript- and TypeScript-based SPAs, we will learn how to build a web service using the **ASP.NET Core Web API**, and then call that web service from the server-side code in our ASP.NET Core websites, and then later, we will call that web service from Blazor WebAssembly components and cross-platform mobile and desktop apps.

There are no formal definitions, but services are sometimes described based on their complexity:

- **Service**: all functionality needed by a client app in one monolithic service.
- **Microservice**: multiple services that each focus on a smaller set of functionalities.
- **Nanoservice**: a single function provided as a service. Unlike services and microservices that are hosted 24/7/365, nanoservices are often inactive until called upon to reduce resources and costs.

New features in ASP.NET Core

Over the past few years, Microsoft has rapidly expanded the capabilities of ASP.NET Core. You should note which .NET platforms are supported, as shown in the following list:

- ASP.NET Core 1.0 to 2.2 runs on either .NET Core or .NET Framework.
- ASP.NET Core 3.0 or later only runs on .NET Core 3.0 or later.

ASP.NET Core 1.0

ASP.NET Core 1.0 was released in June 2016 and focused on implementing a minimum API suitable for building modern cross-platform web apps and services for Windows, macOS, and Linux.

ASP.NET Core 1.1

ASP.NET Core 1.1 was released in November 2016 and focused on bug fixes and general improvements to features and performance.

ASP.NET Core 2.0

ASP.NET Core 2.0 was released in August 2017 and focused on adding new features such as Razor Pages, bundling assemblies into a `Microsoft.AspNetCore.All` metapackage, targeting .NET Standard 2.0, providing a new authentication model, and performance improvements.

The biggest new features introduced with ASP.NET Core 2.0 are ASP.NET Core Razor Pages, which is covered in *Chapter 13*, *Building Websites Using ASP.NET Core Razor Pages*, and ASP.NET Core OData support.

ASP.NET Core 2.1

ASP.NET Core 2.1 was released in May 2018 and was a **Long Term Support** (**LTS**) release, meaning it was supported for three years until August 21, 2021 (LTS designation was not officially assigned to it until August 2018 with version 2.1.3).

It focused on adding new features such as **SignalR** for real-time communication, **Razor class libraries** for reusing web components, **ASP.NET Core Identity** for authentication, and better support for HTTPS and the European Union's **General Data Protection Regulation** (**GDPR**), including the topics listed in the following table:

Feature	Chapter	Topic
Razor class libraries	13	Using Razor class libraries
GDPR support	14	Creating and exploring an ASP.NET Core MVC website
Identity UI library and scaffolding	14	Exploring an ASP.NET Core MVC website
Integration tests	14	Testing an ASP.NET Core MVC website
`[ApiController]` , `ActionResult<T>`	15	Creating an ASP.NET Core Web API project
Problem details	15	Implementing a Web API controller
`IHttpClientFactory`	15	Configuring HTTP clients using HttpClientFactory

ASP.NET Core 2.2

ASP.NET Core 2.2 was released in December 2018 and focused on improving the building of RESTful HTTP APIs, updating the project templates to Bootstrap 4 and Angular 6, an optimized configuration for hosting in Azure, and performance improvements, including the topics listed in the following table:

Feature	Chapter	Topic
HTTP/2 in Kestrel	13	Classic ASP.NET versus modern ASP.NET Core
In-process hosting model	13	Creating an ASP.NET Core project
Endpoint routing	13	Understanding endpoint routing
Health Check API	15	Implementing a health check API
Open API analyzers	15	Implementing Open API analyzers and conventions

ASP.NET Core 3.0

ASP.NET Core 3.0 was released in September 2019 and focused on fully leveraging .NET Core 3.0 and .NET Standard 2.1, which meant it could not support .NET Framework, and it added useful refinements, including the topics listed in the following table:

Feature	Chapter	Topic
Static assets in Razor class libraries	13	Using Razor class libraries
New options for MVC service registration	14	Understanding ASP.NET Core MVC startup
Blazor Server	16	Building components using Blazor Server

ASP.NET Core 3.1

ASP.NET Core 3.1 was released in December 2019 and is an LTS release, meaning it will be supported until December 3, 2022. It focused on refinements like partial class support for Razor components and a new `<component>` tag helper.

Blazor WebAssembly 3.2

Blazor WebAssembly 3.2 was released in May 2020. It was a Current release, meaning that projects had to be upgraded to the .NET 5 version within three months of the .NET 5 release, that is, by February 10, 2021. Microsoft finally delivered on the promise of full-stack web development with .NET, and both Blazor Server and Blazor WebAssembly are covered in *Chapter 16*, *Building User Interfaces Using Blazor*.

ASP.NET Core 5.0

ASP.NET Core 5.0 was released in November 2020 and focused on bug fixes, performance improvements using caching for certificate authentication, HPACK dynamic compression of HTTP/2 response headers in Kestrel, nullable annotations for ASP.NET Core assemblies, and a reduction in container image sizes, including the topics listed in the following table:

Feature	Chapter	Topic
Extension method to allow anonymous access to an endpoint	15	Securing web services
JSON extension methods for `HttpRequest` and `HttpResponse`	15	Getting customers as JSON in the controller

ASP.NET Core 6.0

ASP.NET Core 6.0 was released in November 2021 and focused on productivity improvements like minimizing code to implement basic websites and services, support for .NET Hot Reload, and new hosting options for Blazor, like hybrid apps using .NET MAUI, including the topics listed in the following table:

Feature	Chapter	Topic
New empty web project template	13	Understanding the empty web template
Minimal APIs	15	Implementing minimal Web APIs
Blazor WebAssembly AOT	16	Enabling Blazor WebAssembly ahead-of-time compilation

ASP.NET Core 7.0

ASP.NET Core 7.0 was released in November 2022 and focused on ???, including the topics listed in the following table:

Feature	Chapter	Topic
HTTP/3 support	13	Enabling HTTP/3 support
HTTP/3 client support	15	Enabling HTTP/3 support for HttpClient

Structuring projects

How should you structure your projects? So far, we have built small individual console apps to illustrate language or library features. In the rest of this book, we will build multiple projects using different technologies that work together to provide a single solution.

With large, complex solutions, it can be difficult to navigate amongst all the code. So, the primary reason to structure your projects is to make it easier to find components. It is good to have an overall name for your solution or workspace that reflects the application or solution.

We will build multiple projects for a fictional company named **Northwind**. We will name the solution or workspace `PracticalApps` and use the name `Northwind` as a prefix for all the project names.

There are many ways to structure and name projects and solutions, for example, using a folder hierarchy as well as a naming convention. If you work in a team, make sure you know how your team does it.

Structuring projects in a solution or workspace

It is good to have a naming convention for your projects in a solution or workspace so that any developer can tell what each one does instantly. A common choice is to use the type of project, for example, class library, console app, website, and so on, as shown in the following table:

Name	Description
`Northwind.Common`	A class library project for common types like interfaces, enums, classes, records, and structs, used across multiple projects.
`Northwind.Common.EntityModels`	A class library project for common EF Core entity models. Entity models are often used on both the server and client side, so it is best to separate dependencies on specific database providers.
`Northwind.Common.DataContext`	A class library project for the EF Core database context with dependencies on specific database providers.
`Northwind.Web`	An ASP.NET Core project for a simple website that uses a mixture of static HTML files and dynamic Razor Pages.
`Northwind.Razor.Component`	A class library project for Razor Pages used in multiple projects.
`Northwind.Mvc`	An ASP.NET Core project for a complex website that uses the MVC pattern and can be more easily unit tested.
`Northwind.WebApi`	An ASP.NET Core project for an HTTP API service. A good choice for integrating with websites because they can use any JavaScript library or Blazor to interact with the service.
`Northwind.BlazorServer`	An ASP.NET Core Blazor Server project.
`Northwind.BlazorWasm.Client`	An ASP.NET Core Blazor WebAssembly client-side project.
`Northwind.BlazorWasm.Server`	An ASP.NET Core Blazor WebAssembly server-side project.

Building an entity data model for the Northwind database

Practical applications usually need to work with data in a relational database or another data store. In this chapter, we will define an entity data model for the Northwind database stored in SQL Server or SQLite. It will be used in most of the apps that we create in subsequent chapters.

The `Northwind4SQLServer.sql` and `Northwind4SQLite.sql` script files are different. The script for SQL Server creates 13 tables as well as related views and stored procedures. The script for SQLite is a simplified version that only creates 10 tables because SQLite does not support as many features. The main projects in this book only need those 10 tables so you can complete every task in this book with either database.

Instructions to install SQLite can be found in *Chapter 10*, *Working with Data Using Entity Framework Core*. In that chapter, you will also find instructions for installing the `dotnet-ef` tool, which you will use to scaffold an entity model from an existing database.

Instructions to install SQL Server can be found in the GitHub repository for the book at the following link: https://github.com/markjprice/cs11dotnet7/blob/main/docs/sql-server/README.md

> **Good Practice**: You should create a separate class library project for your entity data models. This allows easier sharing between backend web servers and frontend desktop, mobile, and Blazor WebAssembly clients.

Creating a class library for entity models using SQLite

You will now define entity data models in a class library so that they can be reused in other types of projects including client-side app models. If you are not using SQL Server, you will need to create this class library for SQLite. If you are using SQL Server, then you can create both a class library for SQLite and one for SQL Server and then switch between them as you choose.

We will automatically generate some entity models using the EF Core command-line tool:

1. Use your preferred code editor to create a new project, as defined in the following list:
   - Project template: **Class Library** / `classlib`
   - Project file and folder: `Northwind.Common.EntityModels.Sqlite`
   - Workspace/solution file and folder: `PracticalApps`
2. In the `Northwind.Common.EntityModels.Sqlite` project, add package references for the SQLite database provider and EF Core design-time support, as shown in the following markup:

```
<ItemGroup>
  <PackageReference
    Include="Microsoft.EntityFrameworkCore.Sqlite" Version="7.0.0" />
  <PackageReference
    Include="Microsoft.EntityFrameworkCore.Design" Version="7.0.0">
    <PrivateAssets>all</PrivateAssets>
    <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAsse
  </PackageReference>
</ItemGroup>
```

Early Access Program: Until the final release of .NET 7, the version numbers of packages will be for previews and release candidates. For example, `7.0.0-preview.1.22109.13`. You can change the version to `7.0.0-*` to automatically upgrade to the latest packages as monthly previews and release candidates are released.

1. Delete the `Class1.cs` file.
2. Build the project.
3. Create the `Northwind.db` file for SQLite by copying the `Northwind4SQLite.sql` file into the `PracticalApps` folder (not the project folder!), and then enter the following command at a command prompt or terminal:

```
sqlite3 Northwind.db -init Northwind4SQLite.sql
```

1. Be patient because this command might take a while to create the database structure, as shown in the following output:

```
-- Loading resources from Northwind4SQLite.sql
SQLite version 3.35.5 2022-04-19 14:49:49
Enter ".help" for usage hints.
sqlite>
```

1. To exit SQLite command mode, press Ctrl + C on Windows or Cmd + D on macOS.
2. Open a command prompt or terminal for the `Northwind.Common.EntityModels.Sqlite` folder.
3. At the command line, generate entity class models for all tables, as shown in the following commands:

```
dotnet ef dbcontext scaffold "Filename=../Northwind.db" Microsoft.EntityFrameworkCore.Sqlite --na
```

Note the following:

- The command to perform: `dbcontext scaffold`
- The connection strings. `"Filename=../Northwind.db"`
- The database provider: `Microsoft.EntityFrameworkCore.Sqlite`
- The namespace: `--namespace Packt.Shared`
- To use data annotations as well as the Fluent API: `--data-annotations`

1. Note the build messages and warnings, as shown in the following output:

```
Build started...
Build succeeded.
To protect potentially sensitive information in your connection string, you should move it out o
```

Improving the class-to-table mapping

The `dotnet-ef` command-line tool generates different code for SQL Server and SQLite because they support different levels of functionality.

For example, SQL Server text columns can have limits to the number of characters. SQLite does not support this. So, `dotnet-ef` will generate validation attributes to ensure `string` properties are limited to a specified number of characters for SQL Server but not for SQLite, as shown in the following code:

```
// SQLite database provider-generated code
[Column(TypeName = "nvarchar (15)")]
public string CategoryName { get; set; } = null!;
// SQL Server database provider-generated code
[StringLength(15)]
public string CategoryName { get; set; } = null!;
```

Neither database provider will mark non-nullable `string` properties as required:

```
// no runtime validation of non-nullable property
public string CategoryName { get; set; } = null!;
// nullable property
public string? Description { get; set; }
// decorate with attribute to perform runtime validation
[Required]
public string CategoryName { get; set; } = null!;
```

We will make some small changes to improve the entity model mapping and validation rules for SQLite:

1. In `Customer.cs`, add a regular expression to validate its primary key value to only allow uppercase Western characters, as shown highlighted in the following code:

```
[Key]
[Column(TypeName = "nchar (5)")]
[RegularExpression("[A-Z]{5}")]
public string CustomerId { get; set; } = null!;
```

1. Activate your code editor's find and replace feature (in Visual Studio 2022, navigate to **Edit** | **Find and Replace** | **Quick Replace**), toggle on **Use Regular Expressions**, and then type a regular expression in the search box, as shown in the following expression:

```
\[Column\(TypeName = "(nchar|nvarchar) \((.*)\)"\)\]
```

1. In the replace box, type a replacement regular expression, as shown in the following expression:

```
$&\n    [StringLength($2)]
```

After the newline character, `\n`, I have included four space characters to indent correctly on my system, which uses two space characters per indentation level. You can insert as many as you wish.

1. Set the find and replace to search files in the current project.

2. Execute the search and replace to replace all, as shown in *Figure 12.2*:

Figure 12.2: Search and replace all matches using regular expressions in Visual Studio 2022

3. Change any date/time properties, for example, in `Employee.cs`, to use a nullable `DateTime` instead of an array of bytes, as shown in the following code:

```
// before
[Column(TypeName = "datetime")]
public byte[]? BirthDate { get; set; }
// after
[Column(TypeName = "datetime")]
public DateTime? BirthDate { get; set; }
```

Use your code editor's find feature to search for `"datetime"` to find all the properties that need changing.

1. Change any `money` properties, for example, in `Order.cs`, to use a nullable `decimal` instead of an array of bytes, as shown in the following code:

```
// before
[Column(TypeName =  "money")]
public byte[]? Freight { get; set; }
// after
[Column(TypeName = "money")]
public decimal? Freight { get; set; }
```

Use your code editor's find feature to search for `"money"` to find all the properties that need changing.

1. In `Product.cs`, make the `Discontinued` property a `bool` instead of an array of bytes, as shown in the following code:

```
[Column(TypeName = "bit")]
public bool Discontinued { get; set; }
```

1. In `Category.cs`, make the `CategoryId` property an `int`, as shown highlighted in the following code:

```
[Key]
public int CategoryId { get; set; }
```

1. In `Category.cs`, make the `CategoryName` property required, as shown highlighted in the following code:

```
[Required]
[Column(TypeName = "nvarchar (15)")]
[StringLength(15)]
public string CategoryName { get; set; }
```

1. In `Customer.cs`, make the `CompanyName` property required, as shown highlighted in the following code:

```
[Required]
[Column(TypeName = "nvarchar (40)")]
[StringLength(40)]
public string CompanyName { get; set; }
```

1. In `Employee.cs`, make the `EmployeeId` property an `int` instead of a `long`.
2. In `Employee.cs`, make the `FirstName` and `LastName` properties required.
3. In `Employee.cs`, make the `ReportsTo` property an `int?` instead of a `long?`.
4. In `EmployeeTerritory.cs`, make the `EmployeeId` property an `int` instead of a `long`.
5. In `EmployeeTerritory.cs`, make the `TerritoryId` property required.
6. In `Order.cs`, make the `OrderId` property an `int` instead of a `long`.
7. In `Order.cs`, decorate the `CustomerId` property with a regular expression to enforce five uppercase characters.
8. In `Order.cs`, make the `EmployeeId` property an `int?` instead of a `long?`.
9. In `Order.cs`, make the `ShipVia` property an `int?` instead of a `long?`.
10. In `OrderDetail.cs`, make the `OrderId` property an `int` instead of a `long`.
11. In `OrderDetail.cs`, make the `ProductId` property an `int` instead of a `long`.
12. In `OrderDetail.cs`, make the `Quantity` property a `short` instead of a `long`.
13. In `Product.cs`, make the `ProductId` property an `int` instead of a `long`.
14. In `Product.cs`, make the `ProductName` property required.
15. In `Product.cs`, make the `SupplierId` and `CategoryId` properties an `int?` instead of a `long?`.
16. In `Product.cs`, make the `UnitsInStock`, `UnitsOnOrder`, and `ReorderLevel` properties a `short?` instead of a `long?`.
17. In `Shipper.cs`, make the `ShipperId` property an `int` instead of a `long`.
18. In `Shipper.cs`, make the `CompanyName` property required.
19. In `Supplier.cs`, make the `SupplierId` property an `int` instead of a `long`.
20. In `Supplier.cs`, make the `CompanyName` property required.
21. In `Territory.cs`, make the `RegionId` property an `int` instead of a `long`.
22. In `Territory.cs`, make the `TerritoryId` and `TerritoryDescription` properties required.

Now that we have a class library for the entity classes, we can create a class library for the database context.

Creating a class library for a Northwind database context

You will now define a database context class library:

1. Add a new project to the solution/workspace, as defined in the following list:
   - Project template: **Class Library** / `classlib`
   - Project file and folder: `Northwind.Common.DataContext.Sqlite`
   - Workspace/solution file and folder: `PracticalApps`
2. In Visual Studio, set the startup project for the solution to the current selection.
3. In Visual Studio Code, select `Northwind.Common.DataContext.Sqlite` as the active OmniSharp project.
4. In the `Northwind.Common.DataContext.Sqlite` project, add a project reference to the `Northwind.Common.EntityModels.Sqlite` project and add a package reference to the EF Core data provider for SQLite, as shown in the following markup:

```
<ItemGroup>
  <PackageReference
    Include="Microsoft.EntityFrameworkCore.SQLite" Version="7.0.0" />
</ItemGroup>
<ItemGroup>
  <ProjectReference Include=
    "..\Northwind.Common.EntityModels.Sqlite\Northwind.Common
.EntityModels.Sqlite.csproj" />
</ItemGroup>
```

Warning! The path to the project reference should not have a line break in your project file.

1. In the `Northwind.Common.DataContext.Sqlite` project, delete the `Class1.cs` class file.

2. Build the `Northwind.Common.DataContext.Sqlite` project.

3. Move the `NorthwindContext.cs` file from the `Northwind.Common.EntityModels.Sqlite` project/folder to the `Northwind.Common.DataContext.Sqlite` project/folder.

   In Visual Studio **Solution Explorer**, if you drag and drop a file between projects it will be copied. If you hold down Shift while dragging and dropping, it will be moved. In Visual Studio Code **EXPLORER**, if you drag and drop a file between projects it will be moved. If you hold down Ctrl while dragging and dropping, it will be copied.

4. In `NorthwindContext.cs`, in the `OnConfiguring` method, remove the compiler `#warning` about the connection string.

   Good Practice: We will override the default database connection string in any projects such as websites that need to work with the Northwind database, so the class derived from `DbContext` must have a constructor with a `DbContextOptions` parameter for this to work, and the generated file does this correctly, as shown in the following code:

```
public NorthwindContext(DbContextOptions<NorthwindContext> options)
  : base(options)
{
}
```

5. In the `OnModelCreating` method, remove all Fluent API statements that call the `ValueGeneratedNever` method to configure primary key properties like `SupplierId` to never generate a value automatically or call the `HasDefaultValueSql` method, as shown in the following code:

```
modelBuilder.Entity<Supplier>(entity =>
{
  entity.Property(e => e.SupplierId).ValueGeneratedNever();
});
```

If we do not remove the configuration like the statements above, then when we add new suppliers, the `SupplierId` value would always be 0 and we would only be able to add one supplier with that value and then all other attempts would throw an exception.

1. For the `Product` entity, tell SQLite that the `UnitPrice` can be converted from `decimal` to `double`. The `OnModelCreating` method should now be much simplified, as shown in the following code:

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
  modelBuilder.Entity<OrderDetail>(entity =>
  {
    entity.HasKey(e => new { e.OrderId, e.ProductId });
    entity.HasOne(d => d.Order)
      .WithMany(p => p.OrderDetails)
      .HasForeignKey(d => d.OrderId)
      .OnDelete(DeleteBehavior.ClientSetNull);
    entity.HasOne(d => d.Product)
      .WithMany(p => p.OrderDetails)
      .HasForeignKey(d => d.ProductId)
      .OnDelete(DeleteBehavior.ClientSetNull);
  });
  modelBuilder.Entity<Product>()
    .Property(product => product.UnitPrice)
    .HasConversion<double>();
  OnModelCreatingPartial(modelBuilder);
}
```

1. In the `Northwind.Common.DataContext.Sqlite` project, add a class named `NorthwindContextExtensions.cs` and modify its contents to define an extension method that adds the Northwind database context to a collection of dependency services, as shown in the following code:

```
using Microsoft.EntityFrameworkCore; // UseSqlite
using Microsoft.Extensions.DependencyInjection; // IServiceCollection
namespace Packt.Shared;
public static class NorthwindContextExtensions
{
  /// <summary>
  /// Adds NorthwindContext to the specified IServiceCollection. Uses the Sqlite database provide
  /// </summary>
  /// <param name="services"></param>
  /// <param name="relativePath">Set to override the default of ".."</param>
  /// <returns>An IServiceCollection that can be used to add more services.</returns>
  public static IServiceCollection AddNorthwindContext(
    this IServiceCollection services, string relativePath = "..")
  {
    string databasePath = Path.Combine(relativePath, "Northwind.db");
    services.AddDbContext<NorthwindContext>(options =>
    {
      options.UseSqlite($"Data Source={databasePath}");
      options.LogTo(WriteLine, // Console
        new[] { Microsoft.EntityFrameworkCore
          .Diagnostics.RelationalEventId.CommandExecuting });
    });
    return services;
  }
}
```

1. Build the two class libraries and fix any compiler errors.

Creating a class library for entity models using SQL Server

To use SQL Server, you will not need to do anything if you already set up the Northwind database in *Chapter 10*, *Working with Data Using Entity Framework Core*. But you will now create the entity models using the `dotnet-ef` tool:

1. Add a new project, as defined in the following list:
   - Project template: **Class Library** / `classlib`
   - Project file and folder: `Northwind.Common.EntityModels.SqlServer`
   - Workspace/solution file and folder: `PracticalApps`
2. In the `Northwind.Common.EntityModels.SqlServer` project, add package references for the SQL Server database provider and EF Core design-time support, as shown in the following markup:

```
<ItemGroup>
  <PackageReference
    Include="Microsoft.EntityFrameworkCore.SqlServer" Version="7.0.0" />
  <PackageReference
    Include="Microsoft.EntityFrameworkCore.Design" Version="7.0.0">
    <PrivateAssets>all</PrivateAssets>
    <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAsse
  </PackageReference>
</ItemGroup>
```

Early Access Program: Until the final release of .NET 7, the version numbers of packages will be for previews and release candidates. For example, `7.0.0-preview.1.22109.13`. You can change the version to `7.0.0-*` to automatically upgrade to the latest packages as monthly previews and release candidates are released.

1. Delete the `Class1.cs` file.
2. Build the project.
3. Open a command prompt or terminal for the `Northwind.Common.EntityModels.SqlServer` folder.
4. At the command line, generate entity class models for all tables, as shown in the following commands:

```
dotnet ef dbcontext scaffold "Data Source=.;Initial Catalog=Northwind;Integrated Security=true;"
```

Note the following:

- The command to perform: `dbcontext scaffold`
- The connection strings. `"Data Source=.;Initial Catalog=Northwind;Integrated Security=true;"`
- The database provider: `Microsoft.EntityFrameworkCore.SqlServer`
- The namespace: `--namespace Packt.Shared`
- To use data annotations as well as the Fluent API: `--data-annotations`

1. In `Customer.cs`, add a regular expression to validate its primary key value to only allow uppercase Western characters, as shown highlighted in the following code:

```
[Key]
[StringLength(5)]
[RegularExpression("[A-Z]{5}")]
public string CustomerId { get; set; } = null!;
```

1. In `Customer.cs`, make the `CustomerId` and `CompanyName` properties required.
2. Add a new project, as defined in the following list:
   - Project template: **Class Library** / `classlib`
   - Project file and folder: `Northwind.Common.DataContext.SqlServer`
   - Workspace/solution file and folder: `PracticalApps`
3. In Visual Studio Code, select `Northwind.Common.DataContext.SqlServer` as the active OmniSharp project.
4. In the `Northwind.Common.DataContext.SqlServer` project, add a project reference to the `Northwind.Common.EntityModels.SqlServer` project and add a package reference to the EF Core data provider for SQL Server, as shown in the following markup:

```
<ItemGroup>
  <PackageReference
    Include="Microsoft.EntityFrameworkCore.SqlServer" Version="7.0.0" />
</ItemGroup>
<ItemGroup>
  <ProjectReference Include=
    "..\Northwind.Common.EntityModels.SqlServer\Northwind.Common
.EntityModels.SqlServer.csproj" />
</ItemGroup>
```

Warning! The path to the project reference should not have a line break in your project file.

1. In the `Northwind.Common.DataContext.SqlServer` project, delete the `Class1.cs` file.
2. Build the `Northwind.Common.DataContext.SqlServer` project.
3. Move the `NorthwindContext.cs` file from the `Northwind.Common.EntityModels.SqlServer` project/folder to the `Northwind.Common.DataContext.SqlServer` project/folder.
4. In the `Northwind.Common.DataContext.SqlServer` project, in `NorthwindContext.cs`, remove the compiler warning about the connection string.
5. the `Northwind.Common.DataContext.SqlServer` project, add a class named `NorthwindContextExtensions.cs`, and modify its contents to define an extension method that adds the Northwind database context to a collection of dependency services, as shown in the following code:

```
using Microsoft.EntityFrameworkCore; // UseSqlServer
using Microsoft.Extensions.DependencyInjection; // IServiceCollection
namespace Packt.Shared;
public static class NorthwindContextExtensions
{
  /// <summary>
  /// Adds NorthwindContext to the specified IServiceCollection. Uses the SqlServer database prov
  /// </summary>
  /// <param name="services"></param>
  /// <param name="connectionString">Set to override the default.</param>
  /// <returns>An IServiceCollection that can be used to add more services.</returns>
  public static IServiceCollection AddNorthwindContext(
    this IServiceCollection services,
    string connectionString = "Data Source=.;Initial Catalog=Northwind;" +
      "Integrated Security=true;MultipleActiveResultsets=true;Encrypt=false")
  {
    services.AddDbContext<NorthwindContext>(options =>
    {
```

```
      options.UseSqlServer(connectionString);
      options.LogTo(WriteLine, // Console
        new[] { Microsoft.EntityFrameworkCore
          .Diagnostics.RelationalEventId.CommandExecuting });
    });
    return services;
  }
}
```

1. Build the two class libraries and fix any compiler errors.

Good Practice: We have provided optional arguments for the `AddNorthwindContext` method so that we can override the hardcoded SQLite database filename path or the SQL Server database connection string. This will allow us more flexibility, for example, to load these values from a configuration file.

Understanding web development

Developing for the web means developing with **Hypertext Transfer Protocol** (**HTTP**), so we will start by reviewing this important foundational technology.

Understanding Hypertext Transfer Protocol

To communicate with a web server, the client, also known as the **user agent**, makes calls over the network using HTTP. As such, HTTP is the technical underpinning of the web. So, when we talk about websites and web services, we mean that they use HTTP to communicate between a client (often a web browser) and a server.

A client makes an HTTP request for a resource, such as a page, uniquely identified by a **Uniform Resource Locator** (**URL**), and the server sends back an HTTP response, as shown in *Figure 12.3*:

Figure 12.3: An HTTP request and response

You can use Google Chrome and other browsers to record requests and responses.

Good Practice: Google Chrome is currently used by about two thirds of website visitors worldwide, and it has powerful, built-in developer tools, so it is a good first choice for testing your websites. Test your websites with Chrome and at least two other browsers, for example, Firefox and Safari for macOS and iPhone. Microsoft Edge switched from using Microsoft's own rendering engine to using Chromium in 2019, so it is less important to test with it. If Microsoft's Internet Explorer is used at all, it tends to mostly be inside organizations for intranets.

Understanding the components of a URL

A **Uniform Resource Locator** (**URL**) is made up of several components:

- **Scheme**: `http` (clear text) or `https` (encrypted).
- **Domain**: For a production website or service, the **top-level domain** (**TLD**) might be `example.com`. You might have subdomains such as `www`, `jobs`, or `extranet`. During development, you typically use `localhost` for all websites and services.
- **Port number**: For a production website or service, `80` for `http`, `443` for `https`. These port numbers are usually inferred from the scheme. During development, other port numbers are commonly used, such as `5000`, `5001`, and so on, to differentiate between websites and services that all use the shared domain `localhost`.
- **Path**: A relative path to a resource, for example, `/customers/germany`.
- **Query string**: A way to pass parameter values, for example, `?country=Germany&searchtext=shoes`.
- **Fragment**: A reference to an element on a web page using its `id`, for example, `#toc`.

URL is a subset of **Uniform Resource Identifier (URI)**. A URL specifies where a resource is located and how to get it. A URI identifies a resource either by URL or **URN (Uniform Resource Name)**.

Assigning port numbers for projects in this book

In this book, we will use the domain `localhost` for all websites and web services, so we will use port numbers to differentiate projects when multiple need to execute at the same time, as shown in the following table:

Project	Description	Port numbers
`Northwind.Web`	ASP.NET Core Razor Pages website	5000 HTTP, 5001 HTTPS
`Northwind.Mvc`	ASP.NET Core MVC website	5000 HTTP, 5001 HTTPS
`Northwind.WebApi`	ASP.NET Core Web API service	5002 HTTPS
`Minimal.WebApi`	ASP.NET Core Web API (minimal)	5003 HTTPS
`Northwind.BlazorServer`	ASP.NET Core Blazor Server	5004 HTTP, 5005 HTTPS
`Northwind.BlazorWasm`	ASP.NET Core Blazor WebAssembly	5006 HTTP, 5007 HTTPS

Using Google Chrome to make HTTP requests

Let's explore how to use Google Chrome to make HTTP requests:

1. Start Google Chrome.
2. Navigate to **More tools** | **Developer tools**.

3. Click the **Network** tab, and Chrome should immediately start recording the network traffic between your browser and any web servers (note the red circle), as shown in *Figure 12.4*:

Figure 12.4: Chrome Developer Tools recording network traffic

4. In Chrome's address box, enter the address of Microsoft's website for learning ASP.NET, as shown in the following URL:

https://dotnet.microsoft.com/learn/aspnet

1. In **Developer Tools**, in the list of recorded requests, scroll to the top and click on the first entry, the row where the **Type** is **document**, as shown in *Figure 12.5*:

Figure 12.5: Recorded requests in Developer Tools

2. On the right-hand side, click on the **Headers** tab, and you will see details about **Request Headers** and **Response Headers**, as shown in *Figure 12.6*:

Figure 12.6: Request and response headers

Note the following aspects:

- **Request Method** is `GET`. Other HTTP methods that you could see here include `POST`, `PUT`, `DELETE`, `HEAD`, and `PATCH`.
- **Status Code** is `200 OK`. This means that the server found the resource that the browser requested and has returned it in the body of the response. Other status codes that you might see in response to a `GET` request include `301 Moved Permanently`, `400 Bad Request`, `401 Unauthorized`, and `404 Not Found`.
- **Request Headers** sent by the browser to the web server include:
- **accept**, which lists what formats the browser accepts. In this case, the browser is saying it understands `HTML`, `XHTML`, `XML`, and some image formats, but it will accept all other files (`*/*`). Default weightings, also known as quality values, are `1.0`. XML is specified with a quality value of `0.9` so it is preferred less than HTML or XHTML. All other file types are given a quality value of `0.8` so are least preferred.
- **accept-encoding**, which lists what compression algorithms the browser understands, in this case, `GZIP`, `DEFLATE`, and `Brotli`.
- **accept-language**, which lists the human languages it would prefer the content to use. In this case, US English, which has a default quality value of `1.0`, then any dialect of English that has an explicitly specified quality value of `0.9`, and then any dialect of Swedish that has an explicitly specified quality value of `0.8`.
- **Response Headers**, `content-encoding` tells me the server has sent back the HTML web page response compressed using the GZIP algorithm because it knows that the client can decompress that format. (This is not visible in *Figure 12.6* because there is not enough space to expand the **Response Headers** section.)

1. Close Chrome.

Understanding client-side web development technologies

When building websites, a developer needs to know more than just C# and .NET. On the client (that is, in the web browser), you will use a combination of the following technologies:

- **HTML5**: This is used for the content and structure of a web page.
- **CSS3**: This is used for the styles applied to elements on the web page.
- **JavaScript**: This is used to code any business logic needed on the web page, for example, validating form input or making calls to a web service to fetch more data needed by the web page.

Although HTML5, CSS3, and JavaScript are the fundamental components of frontend web development, there are many additional technologies that can make frontend web development more productive, including:

- **Bootstrap**, the world's most popular frontend open-source toolkit.
- **SASS** and **LESS**, CSS preprocessors for styling.
- Microsoft's **TypeScript** language for writing more robust code.
- JavaScript libraries such as **Angular**, **jQuery**, **React**, and **Vue**.

All these higher-level technologies ultimately translate or compile to the underlying three core technologies, so they work across all modern browsers.

As part of the build and deploy process, you will likely use technologies such as:

- **Node.js**, a framework for server-side development using JavaScript.
- **Node Package Manager** (**npm**) and **Yarn**, both client-side package managers.
- **Webpack**, a popular module bundler, a tool for compiling, transforming, and bundling website source files.

Practicing and exploring

Explore this chapter's topics with deeper research.

Exercise 12.1 – Explore topics

Use the links on the following page to learn more detail about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-12---introducing-web-development-using-asp-net-core

Summary

In this chapter, you have been introduced to some of the app models and workloads that you can use to build websites and web services using C# and .NET.

You have created two to four class libraries to define an entity data model for working with the Northwind database using either SQLite or SQL Server or both.

In the following six chapters, you will learn the details about how to build the following:

- Simple websites using static HTML pages and dynamic Razor Pages.
- Complex websites using the **Model-View-Controller** (**MVC**) design pattern.
- Web services that can be called by any platform that can make an HTTP request and client websites that call those web services.
- Blazor user interface components that can be hosted on a web server, in the browser, or on hybrid web-native mobile and desktop apps.

13 Building Websites Using ASP.NET Core Razor Pages

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about building websites with a modern HTTP architecture on the server side using Microsoft ASP.NET Core. You will learn about building simple websites using the ASP.NET Core Razor Pages feature introduced with ASP.NET Core 2.0 and the Razor class library feature introduced with ASP.NET Core 2.1.

This chapter will cover the following topics:

- Exploring ASP.NET Core
- Exploring ASP.NET Core Razor Pages
- Using Entity Framework Core with ASP.NET Core
- Using Razor class libraries
- Configuring services and the HTTP request pipeline
- Enabling HTTP/3 support

Exploring ASP.NET Core

We will start by creating an empty ASP.NET Core project and explore who to enable it to serve simple web pages.

Creating an empty ASP.NET Core project

We will create an ASP.NET Core project that will show a list of suppliers from the Northwind database.

The `dotnet` tool has many project templates that do a lot of work for you, but it can be difficult to know which works best for a given situation, so we will start with the empty website project template and then add features step by step so that you can understand all the pieces:

1. Use your preferred code editor to open the `PracticalApps` solution/workspace and then add a new project, as defined in the following list:
   - Project template: **ASP.NET Core Empty [C#]** / `web`
   - Project file and folder: `Northwind.Web`
   - Workspace/solution file and folder: `PracticalApps`
   - For Visual Studio 2022, leave all other options as their defaults, for example, **Configure for HTTPS** selected, and **Enable Docker** cleared.
2. In Visual Studio Code, select `Northwind.Web` as the active `OmniSharp` project.
3. Build the `Northwind.Web` project.
4. Open the `Northwind.Web.csproj` file and note that the project is like a class library except that the SDK is `Microsoft.NET.Sdk.Web`, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
```

```
  </PropertyGroup>
</Project>
```

1. Add an element to globally and statically import the `System.Console` class.
2. If you are using Visual Studio 2022, in **Solution Explorer**, toggle **Show All Files**.
3. Expand the `obj` folder, expand the `Debug` folder, expand the `net7.0` folder, select the `Northwind.Web.GlobalUsings.g.cs` file, and note the implicitly imported namespaces include all the ones for a console app or class library, as well as some ASP.NET Core ones, such as `Microsoft.AspNetCore.Builder`, as shown in the following code:

```
// <autogenerated />
global using global::Microsoft.AspNetCore.Builder;
global using global::Microsoft.AspNetCore.Hosting;
global using global::Microsoft.AspNetCore.Http;
global using global::Microsoft.AspNetCore.Routing;
global using global::Microsoft.Extensions.Configuration;
global using global::Microsoft.Extensions.DependencyInjection;
global using global::Microsoft.Extensions.Hosting;
global using global::Microsoft.Extensions.Logging;
global using global::System;
global using global::System.Collections.Generic;
global using global::System.IO;
global using global::System.Linq;
global using global::System.Net.Http;
global using global::System.Net.Http.Json;
global using global::System.Threading;
global using global::System.Threading.Tasks;
global using static global::System.Console;
```

1. Close the file and collapse the `obj` folder.
2. Open `Program.cs`, and note the following:
   - An ASP.NET Core project is like a top-level console app, with a hidden `<Main>$` method as its entry point that has an argument passed using the name `args`.
   - It calls `WebApplication.CreateBuilder`, which creates a host for the website using defaults for a web host that is then built.
   - The website will respond to all HTTP `GET` requests with plain text: `Hello World!`.
   - The call to the `Run` method is a blocking call, so the hidden `<Main>$` method does not return until the web server stops running.

The contents of `Program.cs`, as shown in the following code:

```
var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();
app.MapGet("/", () => "Hello World!");
app.Run();
```

1. At the bottom of `Program.cs`, add a statement to write a message to the console after the call to the `Run` method and therefore after the web server has stopped, as shown in the following code:

```
WriteLine("This executes after the web server has stopped!");
```

Testing and securing the website

We will now test the functionality of the ASP.NET Core Empty website project. We will also enable encryption of all traffic between the browser and web server for privacy by switching from HTTP to HTTPS. HTTPS is the secure encrypted version of HTTP.

1. For Visual Studio:

   I. In the toolbar, make sure that **Northwind.Web** is selected rather than **IIS Express** or **WSL**, and switch the **Web Browser (Microsoft Edge)** to **Google Chrome**, as shown in *Figure 13.1*:

Figure 13.1: Selecting the Northwind.Web profile with its Kestrel web server in Visual Studio

II. Navigate to **Debug | Start Without Debugging….**
III. The first time you start a secure website, you will be prompted that your project is configured to use SSL, and to avoid warnings in the browser you can choose to trust the self-signed certificate that ASP.NET Core has generated. Click **Yes**.
IV. When you see the **Security Warning** dialog box, click **Yes** again.

2. For Visual Studio Code, in **TERMINAL**, enter the `dotnet run` command.
3. In either Visual Studio's command prompt window or Visual Studio Code's terminal, note the Kestrel web server has started listening on random ports for HTTP and HTTPS, that you can press *Ctrl + C* to shut down the Kestrel web server, and the hosting environment is `Development`, as shown in the following output:

```
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: https://localhost:7251
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: http://localhost:5251
info: Microsoft.Hosting.Lifetime[0]
  Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
  Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
  Content root path: C:\cs11dotnet7\PracticalApps\Northwind.Web
```

- Visual Studio will also start your chosen browser automatically. If you are using Visual Studio Code, you will have to start Chrome manually.

1. Leave the Kestrel web server running.
2. In Chrome, show **Developer Tools**, and click the **Network** tab.
3. Enter the address `http://localhost:5251/` , or whatever port number was assigned to HTTP, and note the response is **Hello World!** in plain text, from the cross-platform Kestrel web server, as shown in *Figure 13.2*:

Figure 13.2: Plain text response from http://localhost:5000/

Chrome also requests a `favicon.ico` file automatically to show in the browser tab but this is missing so it shows as a **404 Not Found** error.

1. Enter the address `https://localhost:5001/` , or whatever port number was assigned to HTTPS, and note if you are not using Visual Studio or if you clicked **No** when prompted to trust the SSL certificate, then the response is a privacy error, as shown in *Figure 13.3*:

Figure 13.3: Privacy error showing SSL encryption has not been enabled with a certificate

You will see this error when you have not configured a certificate that the browser can trust to encrypt and decrypt HTTPS traffic (and so if you do not see this error, it is because you have already configured a certificate).

In a production environment, you would want to pay a company such as Verisign for an SSL certificate because they provide liability protection and technical support.

1. **For Linux Developers**: If you use a Linux variant that cannot create self-signed certificates or you do not mind reapplying for a new certificate every 90 days, then you can get a free certificate from the following link: https://letsencrypt.org

During development, you can tell your OS to trust a temporary development certificate provided by ASP.NET Core.

1. At the command line or terminal, press *Ctrl + C* to shut down the web server, and note the message that is written, as shown highlighted in the following output:

```
info: Microsoft.Hosting.Lifetime[0]
      Application is shutting down...
This executes after the web server has stopped!
C:\cs11dotnet7\PracticalApps\Northwind.Web\bin\Debug\net7.0\Northwind.Web.exe (process 19888) ex:
```

1. If you need to trust a local self-signed SSL certificate, then at the command line or terminal, enter the `dotnet dev-certs https --trust` command, and note the message, **Trusting the HTTPS development certificate was requested**. You might be prompted to enter your password and a valid HTTPS certificate may already be present.

Enabling stronger security and redirect to a secure connection

It is good practice to enable stricter security and automatically redirect requests for HTTP to HTTPS.

1. **Good Practice**: **HTTP Strict Transport Security** (**HSTS**) is an opt-in security enhancement that you should always enable. If a website specifies it and a browser supports it, then it forces all communication over HTTPS and prevents the visitor from using untrusted or invalid certificates.

Let's do that now:

1. In `Program.cs`, after the statement that builds the `app`, add an `if` statement to enable HSTS when *not* in development, as shown in the following code:

```
if (!app.Environment.IsDevelopment())
{
  app.UseHsts();
}
```

1. Add a statement before the call to `MapGet` to redirect HTTP requests to HTTPS, as shown in the following code:

```
app.UseHttpsRedirection();
```

1. Start the `Northwind.Web` website project.
2. If Chrome is still running, close and restart it.
3. In Chrome, show **Developer Tools**, and click the **Network** tab.

4. Enter the address `http://localhost:5251/`, or whatever port number was assigned to HTTP, and note how the server responds with a **307 Temporary Redirect** to port `5001` and that the certificate is now valid and trusted, as shown in *Figure 13.4*:

Figure 13.4: The connection is now secured using a valid certificate and a 307 redirect

5. Close Chrome and shut down the web server.
6. **Good Practice**: Remember to shut down the Kestrel web server whenever you have finished testing a website.

Controlling the hosting environment

In ASP.NET Core 5 and earlier, the project template set a rule to say that while in development mode, any unhandled exceptions will be shown in the browser window for the developer to see the details of the exception, as shown in the following code:

```
if (app.Environment.IsDevelopment())
{
  app.UseDeveloperExceptionPage();
}
```

With ASP.NET Core 6 and later, this code is executed automatically so it is not included in the project template.

How does ASP.NET Core know when we are running in development mode so that the `IsDevelopment` method returns `true`? Let's find out.

ASP.NET Core can read from settings files and environment variables to determine what hosting environment to use, for example, `DOTNET_ENVIRONMENT` or `ASPNETCORE_ENVIRONMENT`.

You can override these settings during local development:

1. In the `Northwind.Web` folder, expand the folder named `Properties`, open the file named `launchSettings.json`, and note the profile named `Northwind.Web` that sets the hosting environment to `Development`, as shown highlighted in the following configuration:

```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:56111",
      "sslPort": 44329
    }
  },
  "profiles": {
```

```
    "Northwind.Web": {
      "commandName": "Project",
      "dotnetRunMessages": "true",
      "launchBrowser": true,
      "applicationUrl": "https://localhost:5001;http://localhost:5000",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

1. For the `applicationUrl`, change the randomly assigned port numbers for `HTTP` to `5000` and `HTTPS` to `5001`.
2. Change the `ASPNETCORE_ENVIRONMENT` to `Production`.
3. If you are using Visual Studio, optionally, change `launchBrowser` to `false` to prevent Visual Studio from automatically launching a browser.
4. Start the website and note the hosting environment is `Production`, as shown in the following output:

```
info: Microsoft.Hosting.Lifetime[0]
  Hosting environment: Production
```

1. Shut down the web server.
2. In `launchSettings.json`, change the environment back to `Development`.

- The `launchSettings.json` file also has a configuration for using IIS as the web server using random port numbers. In this book, we will only be using Kestrel as the web server since it is cross-platform.

Enabling a website to serve static content

A website that only ever returns a single plain text message isn't very useful!

At a minimum, it ought to return static HTML pages, CSS that the web pages will use for styling, and any other static resources, such as images and videos.

By convention, these files should be stored in a directory named `wwwroot` to keep them separate from the dynamically executing parts of your website project.

Creating a folder for static files and a web page

You will now create a folder for your static website resources and a basic index page that uses Bootstrap for styling:

1. In the `Northwind.Web` project/folder, create a folder named `wwwroot`. Note that Visual Studio recognizes it as a special type of folder by giving it a globe icon.
2. Add a new HTML page file to the `wwwroot` folder named `index.xhtml`.
3. In `index.xhtml`, modify its markup to link to CDN-hosted Bootstrap for styling, and use modern good practices such as setting the viewport, as shown in the following markup:

```
<!doctype html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8" />
  <meta name="viewport" content=
```

```
    "width=device-width, initial-scale=1, shrink-to-fit=no" />
  <!-- Bootstrap CSS -->
  <link href=
"https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integ
  <title>Welcome ASP.NET Core!</title>
</head>
<body>
  <div class="container">
    <div class="jumbotron">
      <h1 class="display-3">Welcome to Northwind B2B</h1>
      <p class="lead">We supply products to our customers.</p>
      <hr />
      <h2>This is a static HTML page.</h2>
      <p>Our customers include restaurants, hotels, and cruise lines.</p>
      <p>
        <a class="btn btn-primary"
          href="https://www.asp.net/">Learn more</a>
      </p>
    </div>
  </div>
</body>
</html>
```

Good Practice: To get the latest `<link>` element for Bootstrap, copy and paste it from the documentation at the following link: https://getbootstrap.com/docs/5.1/getting-started/introduction/#starter-template.

Enabling static and default files

If you were to start the website now and enter `http://localhost:5000/index.xhtml` in the address box, the website would return a `404 Not Found` error saying no web page was found. To enable the website to return static files such as `index.xhtml`, we must explicitly configure that feature.

Even if we enable static files, if you were to start the website and enter `http://localhost:5000/` in the address box, the website will return a `404 Not Found` error because the web server does not know what to return by default if no named file is requested.

You will now enable static files, explicitly configure default files, and change the URL path registered that returns the plain text `Hello World!` response:

1. In `Program.cs`, add statements after enabling HTTPS redirection to enable static files and default files, and modify the statement that maps a `GET` request to return the `Hello World!` plain text response to only respond to the URL path `/hello`, as shown highlighted in the following code:

```
app.UseDefaultFiles(); // index.xhtml, default.xhtml, and so on
app.UseStaticFiles();
app.MapGet("/hello", () => "Hello World!");
```

- The call to `UseDefaultFiles` must come before the call to `UseStaticFiles`, or it will not work. You will learn more about the ordering of middleware and endpoint routing at the end of this chapter.

1. Start the website.
2. Start Chrome and show **Developer Tools**.
3. In Chrome, enter `http://localhost:5000/` and note that you are redirected to the HTTPS address on port `5001`, and the `index.xhtml` file is now returned over that secure connection because it is one of the possible default files for this website.
4. In **Developer Tools**, note the request for the Bootstrap stylesheet.
5. In Chrome, enter `http://localhost:5000/hello` and note that it returns the plain text **Hello World!** as before.
6. Close Chrome and shut down the web server.

If all web pages are static, that is, they only get changed manually by a web editor, then our website programming work is complete. But almost all websites need dynamic content, which means a web page that is generated at

runtime by executing code.

The easiest way to do that is to use a feature of ASP.NET Core named **Razor Pages**.

Exploring ASP.NET Core Razor Pages

ASP.NET Core Razor Pages allow a developer to easily mix C# code statements with HTML markup to make the generated web page dynamic. That is why Razor Pages use the `.cshtml` file extension.

By convention, ASP.NET Core looks for Razor Pages in a folder named `Pages`.

Enabling Razor Pages

You will now copy and change the static HTML page into a dynamic Razor Page, and then add and enable the Razor Pages service:

1. In the `Northwind.Web` project folder, create a folder named `Pages`.
2. Copy the `index.xhtml` file into the `Pages` folder. (In Visual Studio, hold down Ctrl while dragging and dropping.)
3. For the file in the `Pages` folder, rename the file extension from `.xhtml` to `.cshtml`.
4. In `index.cshtml`, remove the `<h2>` element that says that this is a static HTML page.
5. In `Program.cs`, before the statement that builds the `app`, add a statement to add ASP.NET Core Razor Pages and its related services, such as model binding, authorization, anti-forgery, views, and tag helpers, as shown in the following code:

```
builder.Services.AddRazorPages();
```

1. In `Program.cs`, before the statement that maps an HTTP `GET` request for the path `/hello`, add a statement to call the `MapRazorPages` method, as shown in the following code:

```
app.MapRazorPages();
```

Adding code to a Razor Page

In the HTML markup of a web page, Razor syntax is indicated by the `@ symbol`. Razor Pages can be described as follows:

- They require the `@page` directive at the top of the file.
- They can optionally have an `@functions` section that defines any of the following:
  - Properties for storing data values, like in a class definition. An instance of that class is automatically instantiated named `Model` that can have its properties set in special methods and you can get the property values in the HTML.
  - Methods named `OnGet`, `OnPost`, `OnDelete`, and so on that execute when HTTP requests are made, such as `GET`, `POST`, and `DELETE`.

Let's now convert the static HTML page into a Razor Page:

1. In the `Pages` folder, in `index.cshtml`, make modifications as specified in the following list:
   - Add the `@page` statement to the top of the file.
   - After the `@page` statement, add an `@functions` statement block.
   - Define a property to store the name of the current day as a `string` value.
   - Define a method to set `DayName` that executes when an `HTTP GET` request is made for the page, as shown in the following code:

```
@page
@functions
{
  public string? DayName { get; set; }
```

```
  public void OnGet()
  {
    Model.DayName = DateTime.Now.ToString("dddd");
  }
}
```

1. Output the day name inside the second HTML paragraph, as shown highlighted in the following markup:

```
<p>It's @Model.DayName! Our customers include restaurants, hotels, and cruise lines.</p>
```

1. Start the website.

2. In Chrome, enter `https://localhost:5001/` and note the current day name is output on the page, as shown in *Figure 13.5*:

Figure 13.5: Welcome to Northwind page showing the current day

3. In Chrome, enter `https://localhost:5001/index.xhtml`, which exactly matches the static filename, and note that it returns the static HTML page as before.
4. In Chrome, enter `https://localhost:5001/hello`, which exactly matches the endpoint route that returns plain text, and note that it returns **Hello World!** as before.
5. Close Chrome and shut down the web server.

Using shared layouts with Razor Pages

Most websites have more than one page. If every page had to contain all of the boilerplate markup that is currently in `index.cshtml`, that would become a pain to manage. So, ASP.NET Core has a feature named **layouts**.

To use layouts, we must create a Razor file to define the default layout for all Razor Pages (and all MVC views) and store it in a `Shared` folder so that it can be easily found by convention. The name of this file can be anything, because we will specify it, but `_Layout.cshtml` is good practice.

We must also create a specially named file to set the default layout file for all Razor Pages (and all MVC views). This file must be named `_ViewStart.cshtml`.

Let's see layouts in action:

1. In the `Pages` folder, add a file named `_ViewStart.cshtml`. (The Visual Studio item template is named **Razor View Start**.)
2. If you are using Visual Studio Code, modify its content, as shown in the following markup:

```
@{
  Layout = "_Layout";
}
```

1. In the `Pages` folder, create a folder named `Shared`.

2. In the `Shared` folder, create a file named `_Layout.cshtml`. (The Visual Studio item template is named **Razor Layout**.)
3. Modify the content of `_Layout.cshtml` (it is similar to `index.cshtml` so you can copy and paste the HTML markup from there), as shown in the following markup:

```
<!doctype html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8" />
  <meta name="viewport" content=
    "width=device-width, initial-scale=1, shrink-to-fit=no" />
  <!-- Bootstrap CSS -->
  <link href=
"https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integ
  <title>@ViewData["Title"]</title>
</head>
<body>
  <div class="container">
    @RenderBody()
    <hr />
    <footer>
      <p>Copyright &copy; 2022 - @ViewData["Title"]</p>
    </footer>
  </div>
  <!-- JavaScript to enable features like carousel -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" int
  @RenderSection("Scripts", required: false)
</body>
</html>
```

While reviewing the preceding markup, note the following:

- `<title>` is set dynamically using server-side code from a dictionary named `ViewData`. This is a simple way to pass data between different parts of an ASP.NET Core website. In this case, the data will be set in a Razor Page class file and then output in the shared layout.
- `@RenderBody()` marks the insertion point for the view being requested.
- A horizontal rule and footer will appear at the bottom of each page.
- At the bottom of the layout is a script to implement some cool features of Bootstrap that we can use later, such as a carousel of images.
- After the `<script>` elements for Bootstrap, we have defined a section named `Scripts` so that a Razor Page can optionally inject additional scripts that it needs.

1. Modify `index.cshtml` to remove all HTML markup except `<div class="jumbotron">` and its contents, and leave the C# code in the `@functions` block that you added earlier.
2. Add a statement to the `OnGet` method to store a page title in the `ViewData` dictionary, and modify the button to navigate to a suppliers page (which we will create in the next section), as shown highlighted in the following markup:

```
@page
@functions
{
  public string? DayName { get; set; }
  public void OnGet()
  {
    ViewData["Title"] = "Northwind B2B";
    Model.DayName = DateTime.Now.ToString("dddd");
  }
}
<div class="jumbotron">
  <h1 class="display-3">Welcome to Northwind B2B</h1>
  <p class="lead">We supply products to our customers.</p>
  <hr />
  <p>It's @Model.DayName! Our customers include restaurants, hotels, and cruise lines.</p>
  <p>
    <a class="btn btn-primary" href="suppliers">
```

```
      Learn more about our suppliers</a>
  </p>
</div>
```

1. Start the website, visit it with Chrome, and note that it has similar behavior as before, although clicking the button for suppliers will give a `404 Not Found` error because we have not created that page yet.

Using code-behind files with Razor Pages

Sometimes, it is better to separate the HTML markup from the data and executable code, so Razor Pages allows you to do this by putting the C# code in **code-behind** class files. They have the same name as the `.cshtml` file but end with `.cshtml.cs`.

You will now create a Razor Page that shows a list of suppliers. In this example, we are focusing on learning about code-behind files. In the next topic, we will load the list of suppliers from a database, but for now, we will simulate that with a hardcoded array of `string` values:

1. In the `Pages` folder, add new files named `Suppliers.cshtml` and `Suppliers.cshtml.cs`. (The Visual Studio item template is named **Razor Page - Empty** and it creates both files.)
2. In `Suppliers.cshtml.cs`, add statements to define a property to store a list of supplier company names and populate it when an HTTP `GET` request for this page is made, as shown in the following code:

```
using Microsoft.AspNetCore.Mvc.RazorPages; // PageModel
namespace Northwind.Web.Pages;
public class SuppliersModel : PageModel
{
  public IEnumerable<string>? Suppliers { get; set; }
  public void OnGet()
  {
    ViewData["Title"] = "Northwind B2B - Suppliers";
    Suppliers = new[]
    {
      "Alpha Co", "Beta Limited", "Gamma Corp"
    };
  }
}
```

While reviewing the preceding markup, note the following:

- `SuppliersModel` inherits from `PageModel`, so it has members such as the `ViewData` dictionary for sharing data. You can right-click on `PageModel` and select **Go To Definition** to see that it has lots more useful features, such as the entire `HttpContext` of the current request.
- `SuppliersModel` defines a property for storing a collection of `string` values named `Suppliers`.
- When an HTTP `GET` request is made for this Razor Page, the `Suppliers` property is populated with some example supplier names from an array of `string` values. Later, we will populate this from the Northwind database.

1. In `Suppliers.cshtml`, modify the markup to render a heading and an HTML table containing the supplier company names, as shown in the following markup:

```
@page
@model Northwind.Web.Pages.SuppliersModel
<div class="row">
  <h1 class="display-2">Suppliers</h1>
  <table class="table">
    <thead class="thead-inverse">
      <tr><th>Company Name</th></tr>
    </thead>
    <tbody>
    @if (Model.Suppliers is not null)
    {
      @foreach(string name in Model.Suppliers)
      {
        <tr><td>@name</td></tr>
```

```
      }
    }
    </tbody>
  </table>
</div>
```

While reviewing the preceding markup, note the following:

- The model type for this Razor Page is set to `SuppliersModel`.
- The page outputs an HTML table with Bootstrap styles.
- The data rows in the table are generated by looping through the `Suppliers` property of `Model` if it is not `null`.

1. Start the website and visit it using Chrome.
2. Click on the button to learn more about suppliers, and note the table of suppliers, as shown in *Figure 13.6*:

Figure 13.6: The table of suppliers loaded from an array of strings

Using Entity Framework Core with ASP.NET Core

Entity Framework Core is a natural way to get real data into a website. In *Chapter 12*, *Introducing Web Development Using ASP.NET Core*, you created two pairs of class libraries: one for the entity models and one for the Northwind database context, for SQL Server and SQLite. You will now use them in your website project.

Configure Entity Framework Core as a service

Functionality such as Entity Framework Core database contexts that are needed by ASP.NET Core should be registered as a dependency service during website startup. The code in the GitHub repository solution and below uses SQLite, but you can easily use SQL Server if you prefer.

Let's see how:

1. In the `Northwind.Web` project, add a project reference to the `Northwind.Common.DataContext` project for either SQLite or SQL Server, as shown in the following markup:

```
<!-- change Sqlite to SqlServer if you prefer -->
<ItemGroup>
  <ProjectReference Include="..\Northwind.Common.DataContext.Sqlite\
Northwind.Common.DataContext.Sqlite.csproj" />
</ItemGroup>
```

Warning! The project reference must go all on one line with no line break.

1. Build the `Northwind.Web` project.
2. In `Program.cs`, import the namespace to work with your entity model types, as shown in the following code:

```
using Packt.Shared; // AddNorthwindContext extension method
```

1. In `Program.cs`, before the statement that builds the `app`, add a statement to register the `Northwind` database context class, as shown in the following code:

```
builder.Services.AddNorthwindContext();
```

1. In the `Northwind.Web` project, in the `Pages` folder, open `Suppliers.cshtml.cs`, and import the namespace for our database context, as shown in the following code:

```
using Packt.Shared; // NorthwindContext
```

1. In the `SuppliersModel` class, add a private field to store the `Northwind` database context and a constructor to set it, as shown in the following code:

```
private NorthwindContext db;
public SuppliersModel(NorthwindContext injectedContext)
{
  db = injectedContext;
}
```

1. Change the `Suppliers` property to contain `Supplier` objects instead of `string` values, as shown highlighted in the following code:

```
public IEnumerable<Supplier>? Suppliers { get; set; }
```

1. In the `OnGet` method, modify the statements to set the `Suppliers` property from the `Suppliers` property of the database context, sorted by country and then company name, as shown highlighted in the following code:

```
public void OnGet()
{
  ViewData["Title"] = "Northwind B2B - Suppliers";
  Suppliers = db.Suppliers
    .OrderBy(c => c.Country).ThenBy(c => c.CompanyName);
}
```

1. Modify the contents of `Suppliers.cshtml` to import the `Packt.Shared` namespace and render multiple columns for each supplier, as shown highlighted in the following markup:

```
@page
@using Packt.Shared
@model Northwind.Web.Pages.SuppliersModel
<div class="row">
  <h1 class="display-2">Suppliers</h1>
  <table class="table">
    <thead class="thead-inverse">
      <tr>
        <th>Company Name</th>
        <th>Country</th>
        <th>Phone</th>
      </tr>
```

```
    </thead>
    <tbody>
    @if (Model.Suppliers is not null)
    {
      @foreach(Supplier s in Model.Suppliers)
      {
        <tr>
          <td>@s.CompanyName</td>
          <td>@s.Country</td>
          <td>@s.Phone</td>
        </tr>
      }
    }
    </tbody>
  </table>
</div>
```

1. Start the website.
2. In **Chrome**, enter `https://localhost:5001/`.
3. Click **Learn more about our suppliers** and note that the supplier table now loads from the database and the data is sorted first by country and then by company name, as shown in *Figure 13.7*:

Figure 13.7: The suppliers table loaded from the Northwind database

Manipulating data using Razor Pages

You will now add functionality to insert a new supplier.

Enabling a model to insert entities

First, you will modify the supplier model so that it responds to `HTTP POST` requests when a visitor submits a form to insert a new supplier:

1. In the `Northwind.Web` project, in the `Pages` folder, open `Suppliers.cshtml.cs` and import the following namespace:

```
using Microsoft.AspNetCore.Mvc; // [BindProperty], IActionResult
```

1. In the `SuppliersModel` class, add a property to store a single supplier and a method named `OnPost` that adds the supplier to the `Suppliers` table in the Northwind database if its model is valid, as shown in the following code:

```
[BindProperty]
public Supplier? Supplier { get; set; }
public IActionResult OnPost()
{
  if ((Supplier is not null) && ModelState.IsValid)
  {
    db.Suppliers.Add(Supplier);
    db.SaveChanges();
    return RedirectToPage("/suppliers");
  }
  else
  {
    return Page(); // return to original page
  }
}
```

While reviewing the preceding code, note the following:

- We added a property named `Supplier` that is decorated with the `[BindProperty]` attribute so that we can easily connect HTML elements on the web page to properties in the `Supplier` class.
- We added a method that responds to HTTP `POST` requests. It checks that all property values conform to validation rules on the `Supplier` class entity model (such as `[Required]` and `[StringLength]`) and then adds the supplier to the existing table and saves changes to the database context. This will generate a SQL statement to perform the insert into the database. Then it redirects to the `Suppliers` page so that the visitor sees the newly added supplier.

Defining a form to insert a new supplier

Next, you will modify the Razor Page to define a form that a visitor can fill in and submit to insert a new supplier:

1. In `Suppliers.cshtml` , after the `@model` declaration, add Microsoft common tag helpers so that we can use the tag helper `asp-for` on this Razor Page, as shown in the following markup:

```
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

1. At the bottom of the file, add a form to insert a new supplier, and use the `asp-for` tag helper to bind the `CompanyName` , `Country` , and `Phone` properties of the `Supplier` class to the input box, as shown in the following markup:

```
<div class="row">
  <p>Enter details for a new supplier:</p>
  <form method="POST">
    <div><input asp-for="Supplier.CompanyName"
                placeholder="Company Name" /></div>
    <div><input asp-for="Supplier.Country"
                placeholder="Country" /></div>
    <div><input asp-for="Supplier.Phone"
                placeholder="Phone" /></div>
    <input type="submit" />
  </form>
</div>
```

While reviewing the preceding markup, note the following:

- The `<form>` element with a `POST` method is normal HTML, so an `<input type="submit" />` element inside it will make an HTTP `POST` request back to the current page with values of any other elements inside that form.
- An `<input>` element with a tag helper named `asp-for` enables data binding to the model behind the Razor Page.

1. Start the website and Chrome.
2. Click **Learn more about our suppliers**, scroll down to the bottom of the page, enter `Bob's Burgers`, `USA`, and `(603) 555-4567`, and then click **Submit**.
3. Note that you see a refreshed suppliers table with the new supplier added near the bottom because it is sorted with the USA suppliers.
4. Close Chrome and shut down the web server.

Injecting a dependency service into a Razor Page

If you have a `.cshtml` Razor Page that does not have a code-behind file, then you can inject a dependency service using the `@inject` directive instead of constructor parameter injection, and then directly reference the injected database context using Razor syntax in the middle of the markup.

Let's create a simple example:

1. In the `Pages` folder, add a new file named `Orders.cshtml`. (The Visual Studio item template is named **Razor Page - Empty** and it creates two files. Delete the `.cshtml.cs` file.)
2. In `Orders.cshtml`, write code and markup to output the number of orders in the Northwind database, as shown in the following markup:

```
@page
@using Packt.Shared
@inject NorthwindContext db
@{
  string title = "Orders";
  ViewData["Title"] = $"Northwind B2B - {title}";
}
<div class="row">
  <h1 class="display-2">@title</h1>
  <p>
    There are @db.Orders.Count() orders in the Northwind database.
  </p>
</div>
```

1. Start the website.
2. In the browser address bar, navigate to `/orders` and note that you see that there are 830 orders in the Northwind database.
3. Close Chrome and shut down the web server.

Using Razor class libraries

Everything related to a Razor Page can be compiled into a class library for easier reuse in multiple projects. With ASP.NET Core 3.0 and later, this can include static files such as HTML, CSS, JavaScript libraries, and media assets such as image files. A website can either use the Razor Page's view as defined in the class library or override it.

Disabling compact folders for Visual Studio Code

Before we implement our Razor class library, I want to explain a Visual Studio Code feature that confused some readers of a previous edition because the feature was added after publishing.

The compact folders feature means that nested folders such as `/Areas/MyFeature/Pages/` are shown in a compact form if the intermediate folders in the hierarchy do not contain files, as shown in *Figure 13.8*:

Figure 13.8: Compact folders enabled or disabled

If you would like to disable the Visual Studio Code compact folders feature, complete the following steps:

1. On Windows, navigate to **File | Preferences | Settings**. On macOS, navigate to **Code | Preferences | Settings**.
2. In the **Search** settings box, enter `compact`.
3. Clear the **Explorer: Compact Folders** checkbox, as shown in *Figure 13.9*:

Figure 13.9: Disabling compact folders for Visual Studio Code

4. Close the **Settings** tab.

Creating a Razor class library

Let's create a new Razor class library:

1. Use your preferred code editor to open the `PracticalApps` solution/workspace and then add a new project, as defined in the following list:
   - Project template: **Razor Class Library** / `razorclasslib`
   - Project file and folder: `Northwind.Razor.Employees`
   - Checkbox/switch: **Support pages and views** / `-s`
   - Workspace/solution file and folder: `PracticalApps`

- `-s` is short for the `--support-pages-and-views` switch that enables the class library to use Razor Pages and `.cshtml` file views.

1. In the `Northwind.Razor.Employees` project, add a project reference to the `Northwind.Common.DataContext` project for either SQLite or SQL Server and note the SDK is `Microsoft.NET.Sdk.Razor`, as shown highlighted in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk.Razor">
  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <AddRazorSupportForMvc>true</AddRazorSupportForMvc>
  </PropertyGroup>
  <ItemGroup>
    <FrameworkReference Include="Microsoft.AspNetCore.App" />
  </ItemGroup>
  <!-- change Sqlite to SqlServer if you prefer -->
  <ItemGroup>
    <ProjectReference Include="..\Northwind.Common.DataContext.Sqlite
\Northwind.Common.DataContext.Sqlite.csproj" />
  </ItemGroup>
</Project>
```

1. **Warning!** The project reference must go all on one line with no line break. Also, do not mix our SQLite and SQL Server projects or you will see compiler errors. If you used SQL Server in the `Northwind.Web` project, then you must use SQL Server in the `Northwind.Razor.Employees` project as well.
2. Build the `Northwind.Razor.Employees` project.
3. Expand the `Areas` folder, right-click the `MyFeature` folder, select **Rename**, enter the new name `PacktFeatures`, and press Enter.
4. In the `PacktFeatures` folder, in the `Pages` subfolder, add a new file named `_ViewStart.cshtml`. (The Visual Studio item template is named **Razor View Start**. Or just copy it from the `Northwind.Web` project.)
5. If you are using Visual Studio Code, modify its content to inform this class library that any Razor Pages should look for a layout with the same name as used in the `Northwind.Web` project, as shown in the following markup:

```
@{
  Layout = "_Layout";
}
```

- We do not need to create the `_Layout.cshtml` file in this project. It will use the one in its host project, for example, the one in the `Northwind.Web` project.

1. In the `Pages` subfolder, rename `Page1.cshtml` to `Employees.cshtml`, and rename `Page1.cshtml.cs` to `Employees.cshtml.cs`.
2. In `Employees.cshtml.cs`, define a page model with an array of `Employee` entity instances loaded from the Northwind database, as shown in the following code:

```
using Microsoft.AspNetCore.Mvc.RazorPages; // PageModel
using Packt.Shared; // Employee, NorthwindContext
namespace PacktFeatures.Pages;
public class EmployeesPageModel : PageModel
{
  private NorthwindContext db;
  public EmployeesPageModel(NorthwindContext injectedContext)
  {
    db = injectedContext;
  }
  public Employee[] Employees { get; set; } = null!;
  public void OnGet()
  {
    ViewData["Title"] = "Northwind B2B - Employees";
    Employees = db.Employees.OrderBy(e => e.LastName)
      .ThenBy(e => e.FirstName).ToArray();
```

```
  }
}
```

1. In `Employees.cshtml`, add markup to x, as shown in the following markup:

```
@page
@using Packt.Shared
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@model PacktFeatures.Pages.EmployeesPageModel
<div class="row">
  <h1 class="display-2">Employees</h1>
</div>
<div class="row">
@foreach(Employee employee in Model.Employees)
{
  <div class="col-sm-3">
    <partial name="_Employee" model="employee" />
  </div>
}
</div>
```

While reviewing the preceding markup, note the following:

- We import the `Packt.Shared` namespace so that we can use classes in it such as `Employee`.
- We add support for tag helpers so that we can use the `<partial>` element.
- We declare the `@model` type for this Razor Page to use the page model class that you just defined.
- We enumerate through the `Employees` in the model, outputting each one using a partial view.

Implementing a partial view to show a single employee

The `<partial>` tag helper was introduced in ASP.NET Core 2.1. A partial view is like a piece of a Razor Page. You will create one in the next few steps to render a single employee:

1. In the `Northwind.Razor.Employees` project, in the `Pages` folder, create a `Shared` folder.
2. In the `Shared` folder, create a file named `_Employee.cshtml`. (The Visual Studio item template is named **Razor View - Empty**.)
3. In `_Employee.cshtml`, add markup to output an employee using a Bootstrap card, as shown in the following markup:

```
@model Packt.Shared.Employee
<div class="card border-dark mb-3" style="max-width: 18rem;">
  <div class="card-header">@Model?.LastName, @Model?.FirstName</div>
  <div class="card-body text-dark">
    <h5 class="card-title">@Model?.Country</h5>
    <p class="card-text">@Model?.Notes</p>
  </div>
</div>
```

While reviewing the preceding markup, note the following:

- By convention, the names of partial views start with an underscore.
- If you put a partial view in the `Shared` folder, then it can be found automatically.
- The model type for this partial view is a single `Employee` entity.
- We use Bootstrap card styles to output information about each employee.

Using and testing a Razor class library

You will now reference and use the Razor class library in the website project:

1. In the `Northwind.Web` project, add a project reference to the `Northwind.Razor.Employees` project, as shown in the following markup:

```
<ProjectReference Include=
  "..\Northwind.Razor.Employees\Northwind.Razor.Employees.csproj" />
```

1. In `Pages\index.cshtml`, add a paragraph with a link to the Packt feature employees page after the link to the suppliers page, as shown in the following markup:

```
<p>
  <a class="btn btn-primary" href="packtfeatures/employees">
    Contact our employees
  </a>
</p>
```

1. Start the website, visit the website using Chrome, and click the **Contact our employees** button to see the cards of employees, as shown in *Figure 13.10*:

Figure 13.10: A list of employees from a Razor class library feature

Configuring services and the HTTP request pipeline

Now that we have built a website, we can return to the `Startup` configuration and review how services and the HTTP request pipeline work in more detail.

Understanding endpoint routing

In earlier versions of ASP.NET Core, the routing system and the extendable middleware system did not always work easily together; for example, if you wanted to implement a policy such as CORS in both middleware and MVC. Microsoft has invested in improving routing with a system named **endpoint routing** introduced with ASP.NET Core 2.2.

> **Good Practice**: Endpoint routing replaces the `IRouter`-based routing used in ASP.NET Core 2.1 and earlier. Microsoft recommends every older ASP.NET Core project migrates to endpoint routing if possible.

Endpoint routing is designed to enable better interoperability between frameworks that need routing, such as Razor Pages, MVC, or Web APIs, and middleware that needs to understand how routing affects them, such as localization, authorization, CORS, and so on.

Endpoint routing gets its name because it represents the route table as a compiled tree of endpoints that can be walked efficiently by the routing system. One of the biggest improvements is the performance of routing and action method selection.

It is on by default with ASP.NET Core 2.2 or later if compatibility is set to 2.2 or later. Traditional routes registered using the `MapRoute` method or with attributes are mapped to the new system.

The new routing system includes a link generation service registered as a dependency service that does not need an `HttpContext`.

Configuring endpoint routing

Endpoint routing requires a pair of calls to the `UseRouting` and `UseEndpoints` methods:

- `UseRouting` marks the pipeline position where a routing decision is made.
- `UseEndpoints` marks the pipeline position where the selected endpoint is executed.

Middleware such as localization that runs in between these methods can see the selected endpoint and can switch to a different endpoint if necessary.

Endpoint routing uses the same route template syntax that has been used in ASP.NET MVC since 2010 and the `[Route]` attribute introduced with ASP.NET MVC 5 in 2013. Migration often only requires changes to the `Startup` configuration.

MVC controllers, Razor Pages, and frameworks such as SignalR used to be enabled by a call to `UseMvc` or similar methods, but they are now added inside the `UseEndpoints` method call because they are all integrated into the same routing system along with middleware.

Reviewing the endpoint routing configuration in our project

Review the statements in `Program.cs`, as shown in the following code:

```
using Packt.Shared; // AddNorthwindContext extension method
// configure services
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddRazorPages();
builder.Services.AddNorthwindContext();
var app = builder.Build();
// configure the HTTP pipeline
if (!app.Environment.IsDevelopment())
{
  app.UseHsts();
}
app.UseHttpsRedirection();
app.UseDefaultFiles(); // index.xhtml, default.xhtml, and so on
app.UseStaticFiles();
app.MapRazorPages();
app.MapGet("/hello", () => "Hello World!");
// start the web server
app.Run();
WriteLine("This executes after the web server has stopped!");
```

The web application `builder` registers services that can then be retrieved when the functionality they provide is needed using dependency injection. The naming convention for a method that registers a service is `AddNameOfSservice`. Our code registers two services: Razor Pages and an EF Core database context.

Common methods that register dependency services, including services that combine other method calls that register services, are shown in the following table:

Method	Services that it registers
`AddMvcCore`	Minimum set of services necessary to route requests and invoke controllers. Most websites will need more configuration than this.
`AddAuthorization`	Authentication and authorization services.

`AddDataAnnotations`	MVC data annotations service.
`AddCacheTagHelper`	MVC cache tag helper service.

`AddRazorPages`	Razor Pages service including the Razor view engine. Commonly used in simple website projects. It calls the following additional methods:
	
	`AddMvcCore`
	
	`AddAuthorization`
	
	`AddDataAnnotations`
	
	`AddCacheTagHelper`

`AddApiExplorer`	Web API explorer service.
`AddCors`	CORS support for enhanced security.
`AddFormatterMappings`	Mappings between a URL format and its corresponding media type.

`AddControllers`	Controller services but not services for views or pages. Commonly used in ASP.NET Core Web API projects. It calls the following additional methods:
	
	`AddMvcCore`
	
	`AddAuthorization`
	
	`AddDataAnnotations`
	
	`AddCacheTagHelper`
	
	`AddApiExplorer`
	
	`AddCors`
	
	`AddFormatterMappings`

`AddViews`	Support for `.cshtml` views including default conventions.
`AddRazorViewEngine`	Support for Razor view engine including processing the `@` symbol.

`AddControllersWithViews`	Controller, views, and pages services. Commonly used in ASP.NET Core MVC website projects. It calls the following additional methods:
	
	`AddMvcCore`
	
	`AddAuthorization`
	
	`AddDataAnnotations`
	
	`AddCacheTagHelper`
	
	`AddApiExplorer`

	`AddCors`
	`AddFormatterMappings`
	`AddViews`
	`AddRazorViewEngine`
`AddMvc`	Similar to `AddControllersWithViews` , but you should only use it for backward compatibility.
`AddDbContext<T>`	Your `DbContext` type and its optional `DbContextOptions<TContext>` .
`AddNorthwindContext`	A custom extension method we created to make it easier to register the `NorthwindContext` class for either SQLite or SQL Server based on the project referenced.

You will see more examples of using these extension methods for registering services in the next few chapters when working with MVC and Web API services.

Setting up the HTTP pipeline

After building the web application and its services, the reach statements that configure the HTTP pipeline, through which HTTP requests and responses flow in and out. The pipeline is made up of a connected sequence of delegates that can perform processing and then decide to either return a response themselves or pass processing on to the next delegate in the pipeline. Responses that come back can also be manipulated.

Remember that delegates define a method signature that a delegate implementation can plug into. The delegate for the HTTP request pipeline is simple, as shown in the following code:

```
public delegate Task RequestDelegate(HttpContext context);
```

You can see that the input parameter is an `HttpContext` . This provides access to everything you might need to process the incoming HTTP request, including the URL path, query string parameters, cookies, and user agent.

These delegates are often called **middleware** because they sit in between the browser client and the website or web service.

Middleware delegates are configured using one of the following methods or a custom method that calls them itself:

- `Run` : Adds a middleware delegate that terminates the pipeline by immediately returning a response instead of calling the next middleware delegate.
- `Map` : Adds a middleware delegate that creates a branch in the pipeline when there is a matching request usually based on a URL path like `/hello` .
- `Use` : Adds a middleware delegate that forms part of the pipeline so it can decide if it wants to pass the request to the next delegate in the pipeline and it can modify the request and response before and after the next delegate.

For convenience, there are many extension methods that make it easier to build the pipeline, for example, `UseMiddleware<T>` , where `T` is a class that has:

1. A constructor with a `RequestDelegate` parameter that will be passed the next pipeline component.
2. An `Invoke` method with a `HttpContext` parameter and returns a `Task` .

Summarizing key middleware extension methods

Key middleware extension methods used in our code include the following:

- `UseDeveloperExceptionPage` : Captures synchronous and asynchronous `System.Exception` instances from the pipeline and generates HTML error responses.
- `UseHsts` : Adds middleware for using HSTS, which adds the `Strict-Transport-Security` header.
- `UseRouting` : Adds middleware that defines a point in the pipeline where routing decisions are made and must be combined with a call to `UseEndpoints` where the processing is then executed. This means that for our code, any URL paths that match `/` or `/index` or `/suppliers` will be mapped to Razor Pages and a match on `/hello` will be mapped to the anonymous delegate. Any other URL paths will be passed on to the next delegate for matching, for example, static files. This is why, although it looks like the mapping for Razor Pages and `/hello` happen after static files in the pipeline, they actually take priority because the call to `UseRouting` happens before `UseStaticFiles` .
- `UseHttpsRedirection` : Adds middleware for redirecting HTTP requests to HTTPS, so in our code a request for `http://localhost:5000` would be modified to `https://localhost:5001` .
- `UseDefaultFiles` : Adds middleware that enables default file mapping on the current path, so in our code it would identify files such as `index.xhtml` .
- `UseStaticFiles` : Adds middleware that looks in `wwwroot` for static files to return in the HTTP response.
- `UseEndpoints` : Adds middleware to execute to generate responses from decisions made earlier in the pipeline. Two endpoints are added, as shown in the following sub-list:
  - `MapRazorPages` : Adds middleware that will map URL paths such as `/suppliers` to a Razor Page file in the `/Pages` folder named `suppliers.cshtml` and return the results as the HTTP response.
  - `MapGet` : Adds middleware that will map URL paths such as `/hello` to an inline delegate that writes plain text directly to the HTTP response.

Visualizing the HTTP pipeline

The HTTP request and response pipeline can be visualized as a sequence of request delegates, called one after the other, as shown in the simplified diagram shown in *Figure 13.11*, which excludes some middleware delegates, such as `UseHsts` :

Figure 13.11: The HTTP request and response pipeline

As mentioned before, the `UseRouting` and `UseEndpoints` methods must be used together. Although the code to define the mapped routes such as `/hello` are written in `UseEndpoints` , the decision about whether an incoming

HTTP request URL path matches and therefore which endpoint to execute is made at the `UseRouting` point in the pipeline.

Implementing an anonymous inline delegate as middleware

A delegate can be specified as an inline anonymous method. We will register one that plugs into the pipeline after routing decisions for endpoints have been made.

It will output which endpoint was chosen, as well as handling one specific route: `/bonjour`. If that route is matched, it will respond with plain text, without calling any further into the pipeline:

1. In `Program.cs`, add statements before the call to `UseHttpsRedirection` to use an anonymous method as a middleware delegate, as shown in the following code:

```
app.Use(async (HttpContext context, Func<Task> next) =>
{
  RouteEndpoint? rep = context.GetEndpoint() as RouteEndpoint;
  if (rep is not null)
  {
    WriteLine($"Endpoint name: {rep.DisplayName}");
    WriteLine($"Endpoint route pattern: {rep.RoutePattern.RawText}");
  }
  if (context.Request.Path == "/bonjour")
  {
    // in the case of a match on URL path, this becomes a terminating
    // delegate that returns so does not call the next delegate
    await context.Response.WriteAsync("Bonjour Monde!");
    return;
  }
  // we could modify the request before calling the next delegate
  await next();
  // we could modify the response after calling the next delegate
});
```

1. Start the website.
2. In Chrome, navigate to `https://localhost:5001/`, look at the console output and note that there was a match on an endpoint route `/`, it was processed as `/index`, and the `Index.cshtml` Razor Page was executed to return the response, as shown in the following output:

```
Endpoint name: /index
Endpoint route pattern:
```

1. Navigate to `https://localhost:5001/suppliers` and note that you can see that there was a match on an endpoint route `/Suppliers` and the `Suppliers.cshtml` Razor Page was executed to return the response, as shown in the following output:

```
Endpoint name: /Suppliers
Endpoint route pattern: Suppliers
```

1. Navigate to `https://localhost:5001/index` and note that there was a match on an endpoint route `/index` and the `Index.cshtml` Razor Page was executed to return the response, as shown in the following output:

```
Endpoint name: /index
Endpoint route pattern: index
```

1. Navigate to `https://localhost:5001/index.xhtml` and note that there is no output written to the console because there was no match on an endpoint route but there was a match for a static file, so it was returned as the response.
2. Navigate to `https://localhost:5001/bonjour` and note that there is no output written to the console because there was no match on an endpoint route. Instead, our delegate matched on `/bonjour`, wrote directly to the response stream, and returned with no further processing.
3. Close Chrome and shut down the web server.

Enabling HTTP/3 support

HTTP/3 uses the same request methods like GET and POST, the same status codes like 200 and 404, and the same headers, but encodes them and maintains session state differently because it runs over QUIC rather than the older and less efficient Transmission Control Protocol (TCP).

At the time of writing, HTTP/3 is supported by about 75% of actively used web browsers, including Chromium-based browsers like Chrome, Edge, and Opera. It is also supported by Firefox and Safari on macOS and iOS (although it is disabled by default).

HTTP/3 brings benefits to all internet connected apps but especially mobile because it supports connection migration using UDP with TLS built-in, so the device does not need to reconnect when moving between WiFi and cellular networks. Each frame of data is encrypted separately so it no longer has the head of line blocking problem in HTTP/2 that happens if a TCP packet is lost and therefore all of the streams are blocked until the data can be recovered.

.NET 6 supported HTTP/3 as a preview feature for both clients and servers. .NET 7 delivers a final full support for the following operating systems:

- Windows 11 and Windows Server 2022
- Linux; you can install QUIC support using `sudo apt install libmsquic.`

Early Access Program: You will need to activate preview features in your project file until the final .NET 7 release: `<EnablePreviewFeatures>true</EnablePreviewFeatures>`

Now let's enable HTTP/3 support in the `Northwind.Web` project:

1. In `Program.cs`, import the namespace for working with HTTP protocols, as shown in the following code:

```
using Microsoft.AspNetCore.Server.Kestrel.Core; // HttpProtocols
```

1. In `Program.cs`, add statements before the call to `Build` to enable all three versions of HTTP, as shown in the following code:

```
builder.WebHost.ConfigureKestrel((context, options) =>
{
  options.ListenAnyIP(5001, listenOptions =>
  {
    listenOptions.Protocols = HttpProtocols.Http1AndHttp2AndHttp3;
    listenOptions.UseHttps(); // HTTP/3 requires secure connections
  });
});
```

1. **Good Practice**: You should not only enable HTTP/3 since about 25% of browsers still do not support it or even HTTP/2.
2. In `appSettings.json`, add an entry to show hosting diagnostics, as shown highlighted in the following configuration:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning",
      "Microsoft.AspNetCore.Hosting.Diagnostics": "Information"
    }
```

1. Start the website.
2. In Chrome, view **Developer tools** and select the **Network** tab.

3. Navigate to `https://localhost:5001/`, and note the **Response Headers** include an entry for **alt-svc** with a value of **h3** indicating HTTP/3 support, as shown in *Figure 13.12*:

Figure 13.12: Chrome showing support for HTTP/3

4. In the console or terminal output, note the hosting diagnostics logs, as shown in the following output:

```
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
      Request starting HTTP/3 GET https://localhost:5001/ - -
info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
      Request finished HTTP/3 GET https://localhost:5001/ - - - 200 - text/html;+charset=utf-8 1
warn: Microsoft.AspNetCore.Server.Kestrel[41]
      One or more of the following response headers have been removed because they are invalid f
```

1. Close Chrome and shut down the web server.

You can learn more about .NET support for HTTP/3 at the following links:

- HTTP/3 support in .NET 6: https://devblogs.microsoft.com/dotnet/http-3-support-in-dotnet-6/
- .NET 6 Networking Improvements - HTTP/3 and QUIC: https://devblogs.microsoft.com/dotnet/dotnet-6-networking-improvements/#http-3-and-quic

- Use HTTP/3 with the ASP.NET Core Kestrel web server: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel/http3

Practicing and exploring

Test your knowledge and understanding by answering some questions, get some hands-on practice, and explore this chapter's topics with deeper research.

Exercise 13.1 – Test your knowledge

Answer the following questions:

1. List six method names that can be specific in an HTTP request.
2. List six status codes and their descriptions that can be returned in an HTTP response.
3. In ASP.NET Core, what is the `Program` class used for?
4. What does the acronym HSTS stand for and what does it do?
5. How do you enable static HTML pages for a website?
6. How do you mix C# code into the middle of HTML to create a dynamic page?
7. How can you define shared layouts for Razor Pages?
8. How can you separate the markup from the code-behind in a Razor Page?
9. How do you configure an Entity Framework Core data context for use with an ASP.NET Core website?
10. How can you reuse Razor Pages with ASP.NET Core 2.2 or later?

Exercise 13.2 – Practice building a data-driven web page

Add a Razor Page to the `Northwind.Web` website that enables the user to see a list of customers grouped by country. When the user clicks on a customer record, they then see a page showing the full contact details of that customer, and a list of their orders.

Exercise 13.3 – Practice building web pages for console apps

Reimplement some of the console apps from earlier chapters as Razor Pages, for example, from *Chapter 4, Writing, Debugging, and Testing Functions*, provide a web user interface to output times tables, calculate tax, and generate factorials and the Fibonacci sequence.

Exercise 13.4 – Explore topics

Use the links on the following page to learn more about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-13---building-websites-using-aspnet-core-razor-pages

Summary

In this chapter, you learned:

- About the foundations of web development using HTTP.
- How to build a simple website that returns static files.
- How to use ASP.NET Core Razor Pages with Entity Framework Core to create web pages that are dynamically generated from information in a database.
- How to configure the HTTP request and response pipeline, what the helper extension methods do, and how you can add your own middleware that affects processing.

In the next chapter, you will learn how to build more complex websites using ASP.NET Core MVC, which separates the technical concerns of building a website into models, views, and controllers to make them easier to

manage.

14 Building Websites Using the Model-View-Controller Pattern

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about building websites with a modern HTTP architecture on the server side using Microsoft ASP.NET Core MVC, including configuration, authentication, authorization, routes, request and response pipeline, models, views, and controllers that make up an ASP.NET Core MVC project.

This chapter will cover the following topics:

- Setting up an ASP.NET Core MVC website
- Exploring an ASP.NET Core MVC website
- Customizing an ASP.NET Core MVC website
- Querying a database and using display templates
- Improving scalability using asynchronous tasks

Setting up an ASP.NET Core MVC website

ASP.NET Core Razor Pages are great for simple websites. For more complex websites, it would be better to have a more formal structure to manage that complexity.

This is where the **Model-View-Controller** (**MVC**) design pattern is useful. It uses technologies like Razor Pages, but allows a cleaner separation between technical concerns, as shown in the following list:

- **Models**: Classes that represent the data entities and view models used on the website.
- **Views**: Razor files, that is, `.cshtml` files, that render data in view models into HTML web pages. Blazor uses the `.razor` file extension, but do not confuse them with Razor files!
- **Controllers**: Classes that execute code when an HTTP request arrives at the web server. The controller methods usually create a view model that may contain entity models and passes it to a view to generate an HTTP response to send back to the web browser or other client.

The best way to understand using the MVC design pattern for web development is to see a working example.

Creating an ASP.NET Core MVC website

You will use a project template to create an ASP.NET Core MVC website project that has a database for authenticating and authorizing users. Visual Studio 2022 defaults to using SQL Server LocalDB for the accounts database. Visual Studio Code (or more accurately the `dotnet` CLI tool) uses SQLite by default and you can specify a switch to use SQL Server LocalDB instead.

Let's see it in action:

1. Use your preferred code editor to open the `PracticalApps` solution/workspace and then add an MVC website project with authentication accounts stored in a database, as defined in the following list:
   - Project template: **ASP.NET Core Web App (Model-View-Controller) [C#]** / `mvc`

- Project file and folder: `Northwind.Mvc`
- Workspace/solution file and folder: `PracticalApps`
- **Additional information - Authentication type**: **Individual Accounts** / `--auth Individual`
- For Visual Studio, leave all other options as their defaults.

2. In Visual Studio Code, select `Northwind.Mvc` as the active OmniSharp project.
3. Build the `Northwind.Mvc` project.
4. At the command line or terminal, use the `help` switch to see other options for this project template, as shown in the following command:

```
dotnet new mvc --help
```

1. Note the results, as shown in the following partial output:

```
ASP.NET Core Web App (Model-View-Controller) (C#)
Author: Microsoft
Description: A project template for creating an ASP.NET Core application with example ASP.NET Co
This template contains technologies from parties other than Microsoft, see https://aka.ms/aspnet
```

There are many options, especially related to authentication, as shown in the following table:

Switches	Description
`-au|--auth`	The type of authentication to use: `None` (default): This choice also allows you to disable HTTPS. `Individual` : Individual authentication that stores registered users and their passwords in a database (SQLite by default). We will use this in the project we create for this chapter. `IndividualB2C` : Individual authentication with Azure AD B2C. `SingleOrg` : Organizational authentication for a single tenant. `MultiOrg` : Organizational authentication for multiple tenants. `Windows` : Windows authentication. Mostly useful for intranets.
`-uld|--use-local-db`	Whether to use SQL Server LocalDB instead of SQLite. This option only applies if `--auth Individual` or `--auth IndividualB2C` is specified. The value is an optional bool with a default of `false` .
`-rrc|--razor-runtime-compilation`	Determines if the project is configured to use Razor runtime compilation in Debug builds. This can improve the performance of startup during debugging because it can defer the compilation of Razor views. The value is an optional bool with a default of `false` .
`-f|--framework`	The target framework for the project. Values can be: `net7.0` (default), `net6.0` , or `netcoreapp3.1`

Creating the authentication database for SQL Server LocalDB

If you created the MVC project using Visual Studio 2022, or you used `dotnet new mvc` with the `-uld` or `--use-local-db` switch, then the database for authentication and authorization will be stored in SQL Server LocalDB. But the database does not yet exist. Let's create it now.

At a command prompt or terminal, in the `Northwind.Mvc` folder, enter the command to run database migrations so that the database used to store credentials for authentication is created, as shown in the following command:

```
dotnet ef database update
```

If you created the MVC project using `dotnet new`, then the database for authentication and authorization will be stored in SQLite and the file has already been created named `app.db`.

The connection string for the authentication database is named `DefaultConnection` and it is stored in the `appsettings.json` file in the root folder for the MVC website project.

For SQL Server LocalDB (with a truncated connection string), see the following setting:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=aspnet-Northwind.Mvc-...;Trust
  },
```

For SQLite, see the following setting:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "DataSource=app.db;Cache=Shared"
  },
```

Exploring the default ASP.NET Core MVC website

Let's review the behavior of the default ASP.NET Core MVC website project template:

1. In the `Northwind.Mvc` project, expand the `Properties` folder, open the `launchSettings.json` file, and note the random port numbers (yours will be different) configured for the project for `HTTPS` and `HTTP`, as shown in the following settings:

```
"profiles": {
  "Northwind.Mvc": {
    "commandName": "Project",
    "dotnetRunMessages": true,
    "launchBrowser": true,
    "applicationUrl": "https://localhost:7274;http://localhost:5274",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
```

1. Change the port numbers to `5001` for `HTTPS` and `5000` for `HTTP`, as shown in the following setting:

```
"applicationUrl": "https://localhost:5001;http://localhost:5000",
```

1. Save the changes to the `launchSettings.json` file.
2. Start the website.
3. Start Chrome and open **Developer Tools**.
4. Navigate to `http://localhost:5000/` and note the following, as shown in *Figure 14.1*:
   - Requests for HTTP are automatically redirected to HTTPS on port `5001`.
   - The top navigation menu with links to **Home**, **Privacy**, **Register**, and **Login**. If the viewport width is 575 pixels or less, then the navigation collapses into a hamburger menu.
   - The title of the website, **Northwind.Mvc**, shown in the header and footer:

Figure 14.1: The ASP.NET Core MVC project template website home page

Understanding visitor registration

By default, passwords must have at least one non-alphanumeric character, they must have at least one digit (0-9), and they must have at least one uppercase letter (A-Z). I use `Pa$$w0rd` in scenarios like this when I am just exploring.

The MVC project template follows best practice for **double-opt-in** (**DOI**), meaning that after filling in an email and password to register, an email is sent to the email address, and the visitor must click a link in that email to confirm that they want to register.

We have not yet configured an email provider to send that email, so we must simulate that step:

1. Close the **Developer Tools** pane.
2. In the top navigation menu, click **Register**.
3. Enter an email and password, and then click the **Register** button. (I used `test@example.com` and `Pa$$w0rd`.)
4. Click the link with the text **Click here to confirm your account** and note that you are redirected to a **Confirm email** web page that you could customize.
5. In the top navigation menu, click **Login**, enter your email and password (note that there is an optional checkbox to remember you, and there are links if the visitor has forgotten their password or they want to register as a new visitor), and then click the **Log in** button.
6. Click your email address in the top navigation menu. This will navigate to an account management page. Note that you can set a phone number, change your email address, change your password, enable two-factor authentication (if you add an authenticator app), and download and delete your personal data. This last feature is good for compliance with legal regulations like the European GDPR.
7. Close Chrome and shut down the web server.

Reviewing an MVC website project structure

In your code editor, in Visual Studio **Solution Explorer** (toggle on **Show All Files**) or in Visual Studio Code **EXPLORER**, review the structure of an MVC website project, as shown in *Figure 14.2*:

Figure 14.2: The default folder structure of an ASP.NET Core MVC project

We will look in more detail at some of these parts later, but for now, note the following:

- `Areas` : This folder contains nested folders and a file needed to integrate your website project with **ASP.NET Core Identity**, which is used for authentication.
- `bin` , `obj` : These folders contain temporary files needed during the build process and the compiled assemblies for the project.
- `Controllers` : This folder contains C# classes that have methods (known as actions) that fetch a model and pass it to a view, for example, `HomeController.cs` .
- `Data` : This folder contains Entity Framework Core migration classes used by the ASP.NET Core Identity system to provide data storage for authentication and authorization, for example, `ApplicationDbContext.cs` .
- `Models` : This folder contains C# classes that represent all of the data gathered together by a controller and passed to a view, for example, `ErrorViewModel.cs` .
- `Properties` : This folder contains a configuration file for IIS or IIS Express on Windows and for launching the website during development named `launchSettings.json` . This file is only used on the local development machine and is not deployed to your production website.
- `Views` : This folder contains the `.cshtml` Razor files that combine HTML and C# code to dynamically generate HTML responses. The `_ViewStart` file sets the default layout and `_ViewImports` imports common namespaces used in all views like tag helpers:
  - `Home` : This subfolder contains Razor files for the home and privacy pages.
  - `Shared` : This subfolder contains Razor files for the shared layout, an error page, and two partial views for logging in and validation scripts.
- `wwwroot` : This folder contains static content used by the website, such as CSS for styling, libraries of JavaScript, JavaScript for this website project, and a `favicon.ico` file. You also put images and other static file resources like PDF documents in here. The project template includes Bootstrap and jQuery libraries.

- `app.db` : This is the SQLite database that stores registered visitors. (If you used SQL Server LocalDB, then it will not be needed.)
- `appsettings.json` and `appsettings.Development.json` : These files contain settings that your website can load at runtime, for example, the database connection string for the ASP.NET Core Identity system and logging levels.
- `Northwind.Mvc.csproj` : This file contains project settings like the use of the Web .NET SDK, an entry for SQLite to ensure that the `app.db` file is copied to the website's output folder, and a list of NuGet packages that your project requires, including:
  - `Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore`
  - `Microsoft.AspNetCore.Identity.EntityFrameworkCore`
  - `Microsoft.AspNetCore.Identity.UI`
  - `Microsoft.EntityFrameworkCore.Sqlite` or `Microsoft.EntityFrameworkCore.SqlServer`
  - `Microsoft.EntityFrameworkCore.Tools`
- `Program.cs` : This file defines a hidden `Program` class that contains the `<Main>$` entry point. It builds a pipeline for processing incoming HTTP requests and hosts the website using default options like configuring the Kestrel web server and loading `appsettings` . It adds and configures services that your website needs, for example, ASP.NET Core Identity for authentication, SQLite or SQL Server for identity data storage, and so on, and routes for your application.

Reviewing the ASP.NET Core Identity database

Open `appsettings.json` to find the connection string used for the ASP.NET Core Identity database, as shown highlighted for SQL Server LocalDB in the following markup:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=aspnet-Northwind.Mvc-2F6A1E12-1
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

If you used SQL Server LocalDB for the identity data store, then you can use **Server Explorer** to connect to the database. You can copy and paste the connection string from the `appsettings.json` file, but remove the second backslash between `(localdb)` and `mssqllocaldb` .

If you installed an SQLite tool such as SQLiteStudio, then you can open the SQLite `app.db` database file.

You can then see the tables that the ASP.NET Core Identity system uses to register users and roles, including the `AspNetUsers` table used to store the registered visitor.

Exploring an ASP.NET Core MVC website

Let's walk through the parts that make up a modern ASP.NET Core MVC website.

Understanding ASP.NET Core MVC initialization

Appropriately enough, we will start by exploring the MVC website's default initialization and configuration:

1. Open the `Program.cs` file and note that it uses the top-level program feature (so there is a hidden `Program` class with a `<Main>$` method). This file can be considered to be divided into four important sections from top to bottom. As you review the sections, you might want to add comments to remind yourself of what each section is used for.

.NET 5 and earlier ASP.NET Core project templates used a `Startup` class to separate these parts into separate methods but with .NET 6 and later, Microsoft encourages putting everything in a single `Program.cs` file.

2. The first section imports some namespaces, as shown in the following code:

```
// Section 1 - import namespaces
using Microsoft.AspNetCore.Identity; // IdentityUser
using Microsoft.EntityFrameworkCore; // UseSqlServer, UseSqlite
using Northwind.Mvc.Data; // ApplicationDbContext
```

Remember that by default, many other namespaces are imported using the implicit usings feature of .NET 6 and later. Build the project and then the globally imported namespaces can be found in the following file: `obj\Debug\net7.0\Northwind.Mvc.GlobalUsings.g.cs`.

1. The second section creates and configures a web host builder. It registers an application database context using SQL Server or SQLite with its database connection string loaded from the `appsettings.json` file for its data storage, adds ASP.NET Core Identity for authentication and configures it to use the application database, and adds support for MVC controllers with views, as shown in the following code:

```
// Section 2 - configure the host web server including services
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
var connectionString = builder.Configuration
  .GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
  options.UseSqlServer(connectionString)); // or UseSqlite
builder.Services.AddDatabaseDeveloperPageExceptionFilter();
builder.Services.AddDefaultIdentity<IdentityUser>(options =>
  options.SignIn.RequireConfirmedAccount = true)
  .AddEntityFrameworkStores<ApplicationDbContext>();
builder.Services.AddControllersWithViews();
var app = builder.Build();
```

The `builder` object has two commonly used objects: `Configuration` and `Services`:

- `Configuration` contains merged values from all the places you could set configuration: `appsettings.json`, environment variables, command-line arguments, and so on
- `Services` is a collection of registered dependency services

The call to `AddDbContext` is an example of registering a dependency service. ASP.NET Core implements the **dependency injection** (**DI**) design pattern so that other components like controllers can request needed services through their constructors. Developers register those services in this section of `Program.cs` (or if using a `Startup` class then in its `ConfigureServices` method.)

1. The third section configures the HTTP pipeline through which requests and responses flow in and out. It configures a relative URL path to run database migrations if the website runs in development, or a friendlier error page and HSTS for production. HTTPS redirection, static files, routing, and ASP.NET Identity are enabled, and an MVC default route and Razor Pages are configured, as shown in the following code:

```
// Section 3 -
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
  app.UseMigrationsEndPoint();
}
else
{
  app.UseExceptionHandler("/Home/Error");
  // The default HSTS value is 30 days. You may want to change this for production scenarios, see
  app.UseHsts();
}
app.UseHttpsRedirection();
app.UseStaticFiles();
```

```
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
app.MapControllerRoute(
  name: "default",
  pattern: "{controller=Home}/{action=Index}/{id?}");
app.MapRazorPages();
```

We learned about most of these methods and features in *Chapter 13*, *Building Websites Using ASP.NET Core Razor Pages*.

> **Good Practice**: What does the extension method `UseMigrationsEndPoint` do? You could read the official documentation, but it does not help much. For example, it does not tell us what relative URL path it defines by default: https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.builder.migrationsendpointextensions.usemigrationsendpoint. Luckily, ASP.NET Core is open source, so we can read the source code and discover what it does, at the following link: https://github.com/dotnet/aspnetcore/blob/main/src/Middleware/Diagnostics.EntityFrameworkCore/src/MigrationsEndPointOptions.cs#L18. Get into the habit of exploring the source code for ASP.NET Core to understand how it works.

Apart from the `UseAuthentication` and `UseAuthorization` methods, the most important new method in this section of `Program.cs` is `MapControllerRoute`, which maps a default route for use by MVC. This route is very flexible because it will map to almost any incoming URL, as you will see in the next topic.

Although we will not create any Razor Pages in this chapter, we need to leave the method call that maps Razor Page support because our MVC website uses ASP.NET Core Identity for authentication and authorization, and it uses a Razor Class Library for its user interface components, like visitor registration and login.

1. The fourth and final section has a thread-blocking method call that runs the website and waits for incoming HTTP requests to respond to, as shown in the following code:

```
// Section 4 - start the host web server listening for HTTP requests
app.Run(); // blocking call
```

Understanding the default MVC route

The responsibility of a route is to discover the name of a controller class to instantiate and an action method to execute with an optional `id` parameter to pass into the method that will generate an HTTP response.

A default route is configured for MVC, as shown in the following code:

```
endpoints.MapControllerRoute(
  name: "default",
  pattern: "{controller=Home}/{action=Index}/{id?}");
```

The route pattern has parts in curly brackets `{}` called **segments**, and they are like named parameters of a method. The value of these segments can be any `string`. Segments in URLs are not case-sensitive.

The route pattern looks at any URL path requested by the browser and matches it to extract the name of a `controller`, the name of an `action`, and an optional `id` value (the `?` symbol makes it optional).

If the user hasn't entered these names, it uses defaults of `Home` for the controller and `Index` for the action (the `=` assignment sets a default for a named segment).

The following table contains example URLs and how the default route would work out the names of a controller and action:

URL	Controller	Action	ID
`/`	Home	Index	

`/Muppet`	Muppet	Index
`/Muppet/Kermit`	Muppet	Kermit
`/Muppet/Kermit/Green`	Muppet	Kermit Green
`/Products`	Products	Index
`/Products/Detail`	Products	Detail
`/Products/Detail/3`	Products	Detail 3

Understanding controllers and actions

In MVC, the C stands for *controller*. From the route and an incoming URL, ASP.NET Core knows the name of the controller, so it will then look for a class that is decorated with the `[Controller]` attribute or derives from a class decorated with that attribute, for example, the Microsoft-provided class named `ControllerBase`, as shown in the following code:

```
namespace Microsoft.AspNetCore.Mvc
{
  //
  // Summary:
  // A base class for an MVC controller without view support.
  [Controller]
  public abstract class ControllerBase
  {
...
```

Understanding the ControllerBase class

As you can see in the XML comment, `ControllerBase` does not support views. It is used for creating web services, as you will see in *Chapter 15*, *Building and Consuming Web Services*.

`ControllerBase` has many useful properties for working with the current HTTP context, as shown in the following table:

Property	Description
`Request`	Just the HTTP request. For example, headers, query string parameters, the body of the request as a stream that you can read from, the content type and length, and cookies.
`Response`	Just the HTTP response. For example, headers, the body of the response as a stream that you can write to, the content type and length, status code, and cookies. There are also delegates like `OnStarting` and `OnCompleted` that you can hook a method up to.
`HttpContext`	Everything about the current HTTP context including the request and response, information about the connection, a collection of features that have been enabled on the server with middleware, and a `User` object for authentication and authorization.

Understanding the Controller class

Microsoft provides another class named `Controller` that your classes can inherit from if they do need view support, as shown in the following code:

```
namespace Microsoft.AspNetCore.Mvc
{
  //
  // Summary:
  // A base class for an MVC controller with view support.
  public abstract class Controller : ControllerBase,
    IActionFilter, IFilterMetadata, IAsyncActionFilter, IDisposable
  {
...
```

`Controller` has many useful properties for working with views, as shown in the following table:

Property	Description
`ViewData`	A dictionary that the controller can store key/value pairs in that is accessible in a view. The dictionary's lifetime is only for the current request/response.
`ViewBag`	A dynamic object that wraps the `ViewData` to provide a friendlier syntax for setting and getting dictionary values.
`TempData`	A dictionary that the controller can store key/value pairs in that is accessible in a view. The dictionary's lifetime is for the current request/response and the next request/response for the same visitor session. This is useful for storing a value during an initial request, responding with a redirect, and then reading the stored value in the subsequent request.

`Controller` has many useful methods for working with views, as shown in the following table:

Property	Description
`View`	Returns a `ViewResult` after executing a view that renders a full response, for example, a dynamically generated web page. The view can be selected using a convention or be specified with a string name. A model can be passed to the view.
`PartialView`	Returns a `PartialViewResult` after executing a view that is part of a full response, for example, a dynamically generated chunk of HTML. The view can be selected using a convention or be specified with a string name. A model can be passed to the view.
`ViewComponent`	Returns a `ViewComponentResult` after executing a component that dynamically generates HTML. The component must be selected by specifying its type or its name. An object can be passed as an argument.
`Json`	Returns a `JsonResult` containing a JSON-serialized object. This can be useful for implementing a simple Web API as part of an MVC controller that primarily returns HTML for a human to view.

Understanding the responsibilities of a controller

The responsibilities of a controller are as follows:

- Identify the services that the controller needs to be in a valid state and to function properly in their class constructor(s).
- Use the action name to identify a method to execute.
- Extract parameters from the HTTP request.
- Use the parameters to fetch any additional data needed to construct a view model and pass it to the appropriate view for the client. For example, if the client is a web browser, then a view that renders HTML would be most appropriate. Other clients might prefer alternative renderings, like document formats such as a PDF file or an Excel file, or data formats, like JSON or XML.
- Return the results from the view to the client as an HTTP response with an appropriate status code.

Let's review the controller used to generate the home, privacy, and error pages:

1. Expand the `Controllers` folder
2. Open the file named `HomeController.cs`
3. Note, as shown in the following code, that:
    - Extra namespaces are imported, which I have added comments for to show which types they are needed for.
    - A private read-only field is declared to store a reference to a logger for the `HomeController` that is set in a constructor.
    - All three action methods call a method named `View` and return the results as an `IActionResult` interface to the client.
    - The `Error` action method passes a view model into its view with a request ID used for tracing. The error response will not be cached:

```
using Microsoft.AspNetCore.Mvc; // Controller, IActionResult
using Northwind.Mvc.Models; // ErrorViewModel
using System.Diagnostics; // Activity
namespace Northwind.Mvc.Controllers;
public class HomeController : Controller
{
  private readonly ILogger<HomeController> _logger;
  public HomeController(ILogger<HomeController> logger)
  {
    _logger = logger;
  }
  public IActionResult Index()
  {
    return View();
  }
  public IActionResult Privacy()
  {
    return View();
  }
  [ResponseCache(Duration = 0,
    Location = ResponseCacheLocation.None, NoStore = true)]
  public IActionResult Error()
  {
    return View(new ErrorViewModel { RequestId =
      Activity.Current?.Id ?? HttpContext.TraceIdentifier });
  }
}
```

If the visitor navigates to a path of `/` or `/Home`, then it is the equivalent of `/Home/Index` because those were the default names for controller and action in the default route.

Understanding the view search path convention

The `Index` and `Privacy` methods are identical in implementation, yet they return different web pages. This is because of **conventions**. The call to the `View` method looks in different paths for the Razor file to generate the web page.

Let's deliberately break one of the page names so that we can see the paths searched by default:

1. In the `Northwind.Mvc` project, expand the `Views` folder and then the `Home` folder.
2. Rename the `Privacy.cshtml` file to `Privacy2.cshtml`.
3. Start the website.

4. Start Chrome, navigate to `https://localhost:5001/`, click **Privacy**, and note the paths that are searched for a view to render the web page (including in `Shared` folders for MVC views and Razor Pages), as shown in *Figure 14.3*:

Figure 14.3: An exception showing the default search path for views

5. Close Chrome and shut down the web server.
6. Rename the `Privacy2.cshtml` file back to `Privacy.cshtml`.

You have now seen the view search path convention, as shown in the following list:

- Specific Razor view: `/Views/{controller}/{action}.cshtml`
- Shared Razor view: `/Views/Shared/{action}.cshtml`
- Shared Razor Page: `/Pages/Shared/{action}.cshtml`

Understanding logging

You have just seen that some errors are caught and written to the console. You can write messages to the console in the same way by using the logger.

1. In the `Controllers` folder, in `HomeController.cs`, in the `Index` method, add statements to use the logger to write some messages of various levels to the console, as shown in the following code:

```
_logger.LogError("This is a serious error (not really!)");
_logger.LogWarning("This is your first warning!");
_logger.LogWarning("Second warning!");
_logger.LogInformation("I am in the Index method of the HomeController.");
```

1. Start the `Northwind.Mvc` website project.
2. Start Chrome and navigate to the home page for the website.
3. At the command prompt or terminal, note the messages, as shown in the following output:

```
fail: Northwind.Mvc.Controllers.HomeController[0]
      This is a serious error (not really!)
warn: Northwind.Mvc.Controllers.HomeController[0]
      This is your first warning!
warn: Northwind.Mvc.Controllers.HomeController[0]
      Second warning!
info: Northwind.Mvc.Controllers.HomeController[0]
      I am in the Index method of the HomeController.
```

1. Close Chrome and shut down the web server.

Understanding filters

When you need to add some functionality to multiple controllers and actions, you can use or define your own filters that are implemented as an attribute class.

Filters can be applied at the following levels:

- At the action level by decorating an action method with the attribute. This will only affect the one action method.
- At the controller level by decorating the controller class with the attribute. This will affect all methods of the controller.
- At the global level by adding the attribute type to the `Filters` collection of the `MvcOptions` instance that can be used to configure MVC when calling the `AddControllersWithViews` method, as shown in the following code:

```
builder.Services.AddControllersWithViews(options =>
  {
    options.Filters.Add(typeof(MyCustomFilter));
  });
```

Using a filter to secure an action method

You might want to ensure that one particular action method of a controller class can only be called by members of certain security roles. You do this by decorating the method with the `[Authorize]` attribute, as described in the following list:

- `[Authorize]` : Only allow authenticated (non-anonymous, logged-in) visitors to access this action method.
- `[Authorize(Roles = "Sales,Marketing")]` : Only allow visitors who are members of the specified role(s) to access this action method.

Let's see an example:

1. In `HomeController.cs` , import the `Microsoft.AspNetCore.Authorization` namespace.
2. Add an attribute to the `Privacy` method to only allow access to logged-in users who are members of a group/role named `Administrators` , as shown highlighted in the following code:

```
[Authorize(Roles = "Administrators")]
public IActionResult Privacy()
```

1. Start the website.
2. Click **Privacy** and note that you are redirected to the log in page.
3. Enter your email and password.
4. Click **Log in** and note the message, **Access denied - You do not have access to this resource**.
5. Close Chrome and shut down the web server.

Enabling role management and creating a role programmatically

By default, role management is not enabled in an ASP.NET Core MVC project, so we must first enable it before creating roles, and then we will create a controller that will programmatically create an `Administrators` role (if it does not already exist) and assign our test user to that role:

1. In `Program.cs` , in the setup of ASP.NET Core Identity and its database, add a call to `AddRoles` to enable role management, as shown highlighted in the following code:

```
services.AddDefaultIdentity<IdentityUser>(
  options => options.SignIn.RequireConfirmedAccount = true)
```

```
  .AddRoles<IdentityRole>() // enable role management
  .AddEntityFrameworkStores<ApplicationDbContext>();
```

1. In `Controllers`, add an empty controller class named `RolesController.cs` and modify its contents, as shown in the following code:

```
using Microsoft.AspNetCore.Identity; // RoleManager, UserManager
using Microsoft.AspNetCore.Mvc; // Controller, IActionResult
using static System.Console;
namespace Northwind.Mvc.Controllers;
public class RolesController : Controller
{
  private string AdminRole = "Administrators";
  private string UserEmail = "test@example.com";
  private readonly RoleManager<IdentityRole> roleManager;
  private readonly UserManager<IdentityUser> userManager;
  public RolesController(RoleManager<IdentityRole> roleManager,
    UserManager<IdentityUser> userManager)
  {
    this.roleManager = roleManager;
    this.userManager = userManager;
  }
  public async Task<IActionResult> Index()
  {
    if (!(await roleManager.RoleExistsAsync(AdminRole)))
    {
      await roleManager.CreateAsync(new IdentityRole(AdminRole));
    }
    IdentityUser user = await userManager.FindByEmailAsync(UserEmail);
    if (user == null)
    {
      user = new();
      user.UserName = UserEmail;
      user.Email = UserEmail;
      IdentityResult result = await userManager.CreateAsync(
        user, "Pa$$w0rd");
      if (result.Succeeded)
      {
        WriteLine($"User {user.UserName} created successfully.");
      }
      else
      {
        foreach (IdentityError error in result.Errors)
        {
          WriteLine(error.Description);
        }
      }
    }
    if (!user.EmailConfirmed)
    {
      string token = await userManager
        .GenerateEmailConfirmationTokenAsync(user);
      IdentityResult result = await userManager
        .ConfirmEmailAsync(user, token);
      if (result.Succeeded)
      {
        WriteLine($"User {user.UserName} email confirmed successfully.");
      }
      else
      {
        foreach (IdentityError error in result.Errors)
        {
          WriteLine(error.Description);
        }
      }
    }
    if (!(await userManager.IsInRoleAsync(user, AdminRole)))
    {
      IdentityResult result = await userManager
        .AddToRoleAsync(user, AdminRole);
      if (result.Succeeded)
```

```
      {
        WriteLine($"User {user.UserName} added to {AdminRole} successfully.");
      }
      else
      {
        foreach (IdentityError error in result.Errors)
        {
          WriteLine(error.Description);
        }
      }
    }
    return Redirect("/");
  }
}
```

Note the following:

- Two fields for the name of the role and email of the user.
- The constructor gets and stores the registered user and role manager dependency services.
- If the `Administrators` role does not exist, we use the role manager to create it.
- We try to find a test user by its email, create it if it does not exist, and then assign the user to the `Administrators` role.
- Since the website uses DOI, we must generate an email confirmation token and use it to confirm the new users email address.
- Success messages and any errors are written out to the console.
- You will be automatically redirected to the home page.

1. Start the website.
2. Click **Privacy** and note that you are redirected to the login page.
3. Enter your email and password. (I used `mark@example.com`.)
4. Click **Log in** and note that you are denied access as before.
5. Click **Home**.
6. In the address bar, manually enter `roles` as a relative URL path, as shown in the following link: `https://localhost:5001/roles`.
7. View the success messages written to the console, as shown in the following output:

```
User test@example.com created successfully.
User test@example.com email confirmed successfully.
User test@example.com added to Administrators successfully.
```

1. Click **Logout**, because you must log out and log back in to load your role memberships when they are created after you have already logged in.
2. Try accessing the **Privacy** page again, enter the email for the new user that was programmatically created, for example, `test@example.com`, and their password, and then click **Log in**, and you should now have access.
3. Close Chrome and shut down the web server.

Using a filter to cache a response

To improve response times and scalability, you might want to cache the HTTP response that is generated by an action method by decorating the method with the `[ResponseCache]` attribute.

You control where the response is cached and for how long by setting parameters, as shown in the following list:

- `Duration`: In seconds. This sets the `max-age` HTTP response header measured in seconds. Common choices are one hour (3600 seconds) and one day (86400 seconds).
- `Location`: One of the `ResponseCacheLocation` values, `Any`, `Client`, or `None`. This sets the `cache-control` HTTP response header.
- `NoStore`: If `true`, this ignores `Duration` and `Location` and sets the cache-control HTTP response header to `no-store`.

Let's see an example:

1. In `HomeController.cs`, add an attribute to the `Index` method to cache the response for 10 seconds on the browser or any proxies between the server and browser, as shown highlighted in the following code:

```
[ResponseCache(Duration = 10 /* seconds */,
  Location = ResponseCacheLocation.Any)]
public IActionResult Index()
```

1. In `Views`, in `Home`, open `Index.cshtml`, and after the Welcome heading, add a paragraph to output the current time in long format to include seconds, as shown in the following markup:

```
<p class="alert alert-primary">@DateTime.Now.ToLongTimeString()</p>
```

1. Start the website.
2. Note the time on the home page.
3. Click **Register** so that you leave the home page.
4. Click **Home** and note the time on the home page is the same because a cached version of the page is used.
5. Click **Register**. Wait at least ten seconds.
6. Click **Home** and note the time has now updated.
7. Click **Log in**, enter your email and password, and then click **Log in**.
8. Note the time on the home page.
9. Click **Privacy**.
10. Click **Home** and note the page is not cached.
11. View the console and note the warning message explaining that your caching has been overridden because the visitor is logged in and, in this scenario, ASP.NET Core uses anti-forgery tokens and they should not be cached, as shown in the following output:

```
warn: Microsoft.AspNetCore.Antiforgery.DefaultAntiforgery[8]
      The 'Cache-Control' and 'Pragma' headers have been overridden and set to 'no-cache, no-sto
```

1. Close Chrome and shut down the web server.

Using a filter to define a custom route

You might want to define a simplified route for an action method instead of using the default route.

For example, to show the privacy page currently requires the following URL path, which specifies both the controller and action:

```
https://localhost:5001/home/privacy
```

We could make the route simpler, as shown in the following link:

```
https://localhost:5001/private
```

Let's see how to do that:

1. In `HomeController.cs`, add an attribute to the `Privacy` method to define a simplified route, as shown highlighted in the following code:

```
[Route("private")]
[Authorize(Roles = "Administrators")]
public IActionResult Privacy()
```

1. Start the website.
2. In the address bar, enter the following URL path:

```
https://localhost:5001/private
```

1. Enter your email and password, click **Log in**, and note that the simplified path shows the **Privacy** page.
2. Close Chrome and shut down the web server.

Understanding entity and view models

In MVC, the M stands for *model*. Models represent the data required to respond to a request. There are two types of models commonly used: entity models and view models.

Entity models represent entities in a database like SQL Server or SQLite. Based on the request, one or more entities might need to be retrieved from data storage. Entity models are defined using classes since they might need to change and then be used to update the underlying data store.

All the data that we want to show in response to a request is the **MVC model**, sometimes called a **view model**, because it is a model that is passed into a view for rendering into a response format like HTML or JSON. View models should be immutable, so they are commonly defined using records.

For example, the following HTTP `GET` request might mean that the browser is asking for the product details page for product number 3:

http://www.example.com/products/details/3

The controller would need to use the ID route value 3 to retrieve the entity for that product and pass it to a view that can then turn the model into HTML for display in a browser.

Imagine that when a user comes to our website, we want to show them a carousel of categories, a list of products, and a count of the number of visitors we have had this month.

We will reference the Entity Framework Core entity data model for the Northwind database that you created in *Chapter 12*, *Introducing Web Development Using ASP.NET Core*:

1. In the `Northwind.Mvc` project, add a project reference to `Northwind.Common.DataContext` for either SQLite or SQL Server, as shown in the following markup:

```
<ItemGroup>
  <!-- change Sqlite to SqlServer if you prefer -->
  <ProjectReference Include=
"..\Northwind.Common.DataContext.Sqlite\Northwind.Common.DataContext.Sqlite.csproj" />
</ItemGroup>
```

1. Build the `Northwind.Mvc` project to compile its dependencies.
2. If you are using SQL Server, or might want to switch between SQL Server and SQLite, then in `appsettings.json`, add a connection string for the Northwind database using SQL Server, as shown highlighted in the following markup:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=aspnet-Northwind.Mvc-DC9C4FAF-I
    "NorthwindConnection": "Server=.;Database=Northwind;Trusted_Connection=True;MultipleActiveRe
  },
```

1. In `Program.cs`, import the namespace to work with your entity model types, as shown in the following code:

```
using Packt.Shared; // AddNorthwindContext extension method
```

1. Before the `builder.Build` method call, add statements to load the appropriate connection string and then to register the `Northwind` database context, as shown in the following code:

```
// if you are using SQL Server
string? sqlServerConnection = builder.Configuration
  .GetConnectionString("NorthwindConnection");
if (sqlServerConnection is null)
```

```
{
  Console.WriteLine("SQL Server database connection string is missing!");
}
else
{
  builder.Services.AddNorthwindContext(sqlServerConnection);
}
// if you are using SQLite default is ..\Northwind.db
builder.Services.AddNorthwindContext();
```

1. Add a class file to the `Models` folder and name it `HomeIndexViewModel.cs`.

   Good Practice: Although the `ErrorViewModel` class created by the MVC project template does not follow this convention, I recommend that you use the naming convention `{Controller}{Action}ViewModel` for your view model classes.

2. In `HomeIndexViewModel.cs`, add statements to define a record that has three properties for a count of the number of visitors, and lists of categories and products, as shown in the following code:

```
using Packt.Shared; // Category, Product
namespace Northwind.Mvc.Models;
public record HomeIndexViewModel
(
  int VisitorCount,
  IList<Category> Categories,
  IList<Product> Products
);
```

1. In `HomeController.cs`, import the `Packt.Shared` namespace, as shown in the following code:

```
using Packt.Shared; // NorthwindContext
```

1. Add a field to store a reference to a `Northwind` instance, and initialize it in the constructor, as shown highlighted in the following code:

```
public class HomeController : Controller
{
  private readonly ILogger<HomeController> _logger;
  private readonly NorthwindContext db;
  public HomeController(ILogger<HomeController> logger,
    NorthwindContext injectedContext)
  {
    _logger = logger;
    db = injectedContext;
  }
```

ASP.NET Core will use constructor parameter injection to pass an instance of the `NorthwindContext` database context using the connection string you specified in `Program.cs`.

1. In the `Index` action method, create an instance of the view model for this method, simulating a visitor count using the `Random` class to generate a number between 1 and 1000, and using the `Northwind` database to get lists of categories and products, and then pass the model to the view, as shown in the following code:

```
  HomeIndexViewModel model = new
  (
    VisitorCount: Random.Shared.Next(1, 1001),
    Categories: db.Categories.ToList(),
    Products: db.Products.ToList()
  );
  return View(model); // pass model to view
}
```

Remember the view search convention: when the `View` method is called in a controller's action method, ASP.NET Core MVC looks in the `Views` folder for a subfolder with the same name as the current controller, that is, `Home`. It

then looks for a file with the same name as the current action, that is, `Index.cshtml`. It will also search for views that match the action method name in the `Shared` folder and for Razor Pages in the `Pages` folder.

Understanding views

In MVC, the V stands for *view*. The responsibility of a view is to transform a model into HTML or other formats.

There are multiple **view engines** that could be used to do this. The default view engine is called **Razor**, and it uses the `@` symbol to indicate server-side code execution. The Razor Pages feature introduced with ASP.NET Core 2.0 uses the same view engine and so can use the same Razor syntax.

Let's modify the home page view to render the lists of categories and products:

1. Expand the `Views` folder, and then expand the `Home` folder.
2. Open the `Index.cshtml` file and note the block of C# code wrapped in `@{ }`. This will execute first and can be used to store data that needs to be passed into a shared layout file like the title of the web page, as shown in the following code:

```
@{
  ViewData["Title"] = "Home Page";
}
```

1. Note the static HTML content in the `<div>` element that uses Bootstrap for styling.

Good Practice: As well as defining your own styles, base your styles on a common library, such as Bootstrap, that implements responsive design.

Just as with Razor Pages, there is a file named `_ViewStart.cshtml` that gets executed by the `View` method. It is used to set defaults that apply to all views.

For example, it sets the `Layout` property of all views to a shared layout file, as shown in the following markup:

```
@{
  Layout = "_Layout";
}
```

1. In the `Views` folder, open the `_ViewImports.cshtml` file and note that it imports some namespaces and then adds the ASP.NET Core tag helpers, as shown in the following code:

```
@using Northwind.Mvc
@using Northwind.Mvc.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

1. In the `Shared` folder, open the `_Layout.cshtml` file.
2. Note that the title is being read from the `ViewData` dictionary that was set earlier in the `Index.cshtml` view, as shown in the following markup:

```
<title>@ViewData["Title"] - Northwind.Mvc</title>
```

1. Note the rendering of links to support Bootstrap and a site stylesheet, where `~` means the `wwwroot` folder, as shown in the following markup:

```
<link rel="stylesheet"
  href="~/lib/bootstrap/dist/css/bootstrap.css" />
<link rel="stylesheet" href="~/css/site.css" />
```

1. Note the rendering of a navigation bar in the header, as shown in the following markup:

```
<body>
  <header>
    <nav class="navbar ...">
```

1. Note the rendering of a collapsible `<div>` containing a partial view for logging in and hyperlinks to allow users to navigate between pages using ASP.NET Core tag helpers with attributes like `asp-controller` and `asp-action`, as shown in the following markup:

```
<div class=
  "navbar-collapse collapse d-sm-inline-flex justify-content-between">
  <ul class="navbar-nav flex-grow-1">
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area=""
        asp-controller="Home" asp-action="Index">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark"
        asp-area="" asp-controller="Home"
        asp-action="Privacy">Privacy</a>
    </li>
  </ul>
  <partial name="_LoginPartial" />
</div>
```

The `<a>` elements use tag helper attributes named `asp-controller` and `asp-action` to specify the controller name and action name that will execute when the link is clicked on. If you want to navigate to a feature in a Razor Class Library, like the `employees` component that you created in the previous chapter, then you use `asp-area` to specify the feature name.

1. Note the rendering of the body inside the `<main>` element, as shown in the following markup:

```
<div class="container">
  <main role="main" class="pb-3">
    @RenderBody()
  </main>
</div>
```

The `RenderBody` method injects the contents of a specific Razor view for a page like the `Index.cshtml` file at that point in the shared layout.

1. Note the rendering of `<script>` elements at the bottom of the page so that it does not slow down the display of the page and that you can add your own script blocks into an optional defined section named `scripts`, as shown in the following markup:

```
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js">
</script>
<script src="~/js/site.js" asp-append-version="true"></script>
@await RenderSectionAsync("Scripts", required: false)
```

Understanding asp-append-version

When `asp-append-version` is specified with a `true` value in any element like `<img>` or `<script>` along with a `src` attribute, the Image Tag Helper is invoked (this helper is poorly named because it does not only affect images!).

It works by automatically appending a query string value named `v` that is generated from a hash of the referenced source file, as shown in the following example generated output:

```
<script src="~/js/site.js? v=Kl_dqr9NVtnMdsM2MUg4qthUnWZm5T1fCEimBPWDNgM"></script>
```

If even a single byte within the `site.js` file changes, then its hash value will be different, and therefore if a browser or CDN is caching the script file, then it will bust the cached copy and replace it with the new version.

Customizing an ASP.NET Core MVC website

Now that you've reviewed the structure of a basic MVC website, you will customize and extend it. You have already registered an EF Core model for the `Northwind` database, so the next task is to output some of that data on the home page.

Defining a custom style

The home page will show a list of the 77 products in the Northwind database. To make efficient use of space, we want to show the list in three columns. To do this, we need to customize the stylesheet for the website:

1. In the `wwwroot\css` folder, open the `site.css` file.
2. At the bottom of the file, add a new style that will apply to an element with the `product-columns` ID, as shown in the following code:

```
#product-columns
{
  column-count: 3;
}
```

Setting up the category images

The Northwind database includes a table of eight categories, but they do not have images, and websites look better with some colorful pictures:

1. In the `wwwroot` folder, create a folder named `images`.
2. In the `images` folder, add eight image files named `category1.jpeg`, `category2.jpeg`, and so on, up to `category8.jpeg`.

You can download images from the GitHub repository for this book at the following link: https://github.com/markjprice/cs11dotnet7/tree/master/Assets/Categories

Understanding Razor syntax and expressions

Before we customize the home page view, let's review an example Razor file that has an initial Razor code block that instantiates an order with price and quantity and then outputs information about the order on the web page, as shown in the following markup:

```
@{
  Order order = new()
  {
    OrderId = 123,
    Product = "Sushi",
    Price = 8.49M,
    Quantity = 3
  };
}
<div>Your order for @order.Quantity of @order.Product has a total cost of $@ order.Price * @orde
```

The preceding Razor file would result in the following incorrect output:

```
Your order for 3 of Sushi has a total cost of $8.49 * 3
```

Although Razor markup can include the value of any single property using the `@object.property` syntax, you should wrap expressions in parentheses, as shown in the following markup:

```
<div>Your order for @order.Quantity of @order.Product has a total cost of $@ (order.Price * orde
```

The preceding Razor expression results in the following correct output:

```
Your order for 3 of Sushi has a total cost of $25.47
```

Defining a typed view

To improve the IntelliSense when writing a view, you can define what type the view can expect using an `@model` directive at the top:

1. In the `Views\Home` folder, open `Index.cshtml`.
2. At the top of the file, add a statement to set the model type to use the `HomeIndexViewModel`, as shown in the following code:

```
@model HomeIndexViewModel
```

Now, whenever we type `Model` in this view, your code editor will know the correct type for the model and will provide IntelliSense for it.

While entering code in a view, remember the following:

- Declare the type for the model, use `@model` (with a lowercase m).
- Interact with the instance of the model, use `@Model` (with an uppercase M).

Let's continue customizing the view for the home page.

1. In the initial Razor code block, add a statement to declare a `string` variable for the current item and under the existing `<div>` element add new markup to output categories in a carousel and products as an unordered list, as shown in the following markup:

```
@using Packt.Shared
@model HomeIndexViewModel
@{
  ViewData["Title"] = "Home Page";
  string currentItem = "";
}
<div class="text-center">
  <h1 class="display-4">Welcome</h1>
  <p class="alert alert-primary">@DateTime.Now.ToLongTimeString()</p>
</div>
@if (Model is not null)
{
<div id="categories" class="carousel slide" data-bs-ride="carousel"
     data-bs-interval="3000" data-keyboard="true">
  <ol class="carousel-indicators">
  @for (int c = 0; c < Model.Categories.Count; c++)
  {
    if (c == 0)
    {
      currentItem = "active";
    }
    else
    {
      currentItem = "";
    }
    <li data-bs-target="#categories" data-slide-to="@c"
        class="@currentItem"></li>
  }
  </ol>
  <div class="carousel-inner">
  @for (int c = 0; c < Model.Categories.Count; c++)
  {
    if (c == 0)
    {
      currentItem = "active";
    }
    else
    {
      currentItem = "";
    }
    <div class="carousel-item @currentItem">
```

```
      <img class="d-block w-100" src=
        "~/images/category@(Model.Categories[c].CategoryId).jpeg"
        alt="@Model.Categories[c].CategoryName" />
      <div class="carousel-caption d-none d-md-block">
        <h2>@Model.Categories[c].CategoryName</h2>
        <h3>@Model.Categories[c].Description</h3>
        <p>
          <a class="btn btn-primary"
            href="/category/@Model.Categories[c].CategoryId">View</a>
        </p>
      </div>
    </div>
  }
  </div>
  <a class="carousel-control-prev" href="#categories"
    role="button" data-bs-slide="prev">
    <span class="carousel-control-prev-icon"
      aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#categories"
    role="button" data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
}
<div class="row">
  <div class="col-md-12">
    <h1>Northwind</h1>
    <p class="lead">
      We have had @Model?.VisitorCount visitors this month.
    </p>
    @if (Model is not null)
    {
    <h2>Products</h2>
    <div id="product-columns">
      <ul class="list-group">
      @foreach (Product p in @Model.Products)
      {
        <li class="list-group-item d-flex justify-content-between align-items-start">
          <a asp-controller="Home" asp-action="ProductDetail"
             asp-route-id="@p.ProductId" class="btn btn-outline-primary">
            <div class="ms-2 me-auto">@p.ProductName</div>
            <span class="badge bg-primary rounded-pill">
              @(p.UnitPrice is null ? "zero" : p.UnitPrice.Value.ToString("C"))
            </span>
          </a>
        </li>
      }
      </ul>
    </div>
    }
  </div>
</div>
```

While reviewing the preceding Razor markup, note the following:

- It is easy to mix static HTML elements such as `<ul>` and `<li>` with C# code to output the carousel of categories and the list of product names.
- The `<div>` element with the `id` attribute of `product-columns` will use the custom style that we defined earlier, so all the content in that element will display in three columns.
- The `<img>` element for each category uses parentheses around a Razor expression to ensure that the compiler does not include the `.jpeg` as part of the expression, as shown in the following markup:
  `"~/images/category@(Model.Categories[c].CategoryID).jpeg"`
- The `<a>` elements for the product links use tag helpers to generate URL paths. Clicks on these hyperlinks will be handled by the `HomeController` and its `ProductDetail` action method. This action method does not exist yet, but you will add it later in this chapter. The ID of the product is passed as a route segment named

`id`, as shown in the following URL path for Ipoh Coffee: `https://localhost:5001/Home/ProductDetail/43`.

Let's see the result of our customized home page:

1. Start the `Northwind.Mvc` website project.
2. Note the home page has a rotating carousel showing categories, a random number of visitors, and a list of products in three columns, as shown in *Figure 14.4*:

Figure 14.4: The updated Northwind MVC website home page

For now, clicking on any of the categories or product links gives **404 Not Found** errors, so let's see how we can implement pages that use the parameters passed to them to see the details of a product or category.

1. Close Chrome and shut down the web server.

Passing parameters using a route value

One way to pass a simple parameter is to use the `id` segment defined in the default route:

1. In `HomeController.cs`, add an action method named `ProductDetail`, as shown in the following code:

```
public IActionResult ProductDetail(int? id)
{
  if (!id.HasValue)
  {
    return BadRequest("You must pass a product ID in the route, for example, /Home/ProductDetail/
  }
  Product? model = db.Products
    .SingleOrDefault(p => p.ProductId == id);
  if (model is null)
  {
    return NotFound($"ProductId {id} not found.");
  }
  return View(model); // pass model to view and then return result
}
```

Note the following:

- This method uses a feature of ASP.NET Core called **model binding** to automatically match the `id` passed in the route to the parameter named `id` in the method.
- Inside the method, we check to see whether `id` does not have a value, and if so, we call the `BadRequest` method to return a `400` status code with a custom message explaining the correct URL path format.
- Otherwise, we can connect to the database and try to retrieve a product using the `id` value.
- If we find a product, we pass it to a view; otherwise, we call the `NotFound` method to return a `404` status code and a custom message explaining that a product with that ID was not found in the database.

1. In the `Views/Home` folder, add a new file named `ProductDetail.cshtml`. (In Visual Studio, the item template is named **Razor View - Empty**.)
2. Modify the contents, as shown in the following markup:

```
@model Packt.Shared.Product
@{
  ViewData["Title"] = "Product Detail - " + Model.ProductName;
}
<h2>Product Detail</h2>
<hr />
<div>
  <dl class="dl-horizontal">
    <dt>Product Id</dt>
    <dd>@Model.ProductId</dd>
    <dt>Product Name</dt>
    <dd>@Model.ProductName</dd>
    <dt>Category Id</dt>
    <dd>@Model.CategoryId</dd>
    <dt>Unit Price</dt>
    <dd>@Model.UnitPrice.Value.ToString("C")</dd>
    <dt>Units In Stock</dt>
    <dd>@Model.UnitsInStock</dd>
  </dl>
</div>
```

1. Start the `Northwind.Mvc` project.
2. When the home page appears with the list of products, click on one of them, for example, the second product, **Chang**.

3. Note the URL path in the browser's address bar, the page title shown in the browser tab, and the product details page, as shown in *Figure 14.5*:

Figure 14.5: The product detail page for Chang

4. View Developer tools.
5. Edit the URL in the address box of Chrome to request a product ID that does not exist, like 99, and note the 404 Not Found status code and custom error response.
6. Close Chrome and shut down the web server.

Understanding model binders in more detail

Model binders are powerful, and the default one does a lot for you. After the default route identifies a controller class to instantiate and an action method to call, if that method has parameters, then those parameters need to have values set.

Model binders do this by looking for parameter values passed in the HTTP request as any of the following types of parameters:

- **Route parameter**, like `id` as we did in the previous section, as shown in the following URL path: `/Home/ProductDetail/2`
- **Query string parameter**, as shown in the following URL path: `/Home/ProductDetail?id=2`
- **Form parameter**, as shown in the following markup:

```
<form action="post" action="/Home/ProductDetail">
  <input type="text" name="id" value="2" />
  <input type="submit" />
</form>
```

Model binders can populate almost any type:

- Simple types, like `int`, `string`, `DateTime`, and `bool`.
- Complex types defined by `class`, `record`, or `struct`.
- Collection types, like arrays and lists.

Let's create a somewhat artificial example to illustrate what can be achieved using the default model binder:

1. In the `Models` folder, add a new file named `Thing.cs`.
2. Modify the contents to define a record with two properties for a nullable integer named `Id` and a `string` named `Color`, as shown in the following code:

```
namespace Northwind.Mvc.Models;
public record Thing(int? Id, string? Color);
```

1. In `HomeController`, add two new action methods, one to show a page with a form and one to display a thing with a parameter using your new model type, as shown in the following code:

```
public IActionResult ModelBinding()
{
  return View(); // the page with a form to submit
}
public IActionResult ModelBinding(Thing thing)
{
  return View(thing); // show the model bound thing
}
```

1. In the `Views\Home` folder, add a new file named `ModelBinding.cshtml`.
2. Modify its contents, as shown in the following markup:

```
@model Thing
@{
  ViewData["Title"] = "Model Binding Demo";
}
<h1>@ViewData["Title"]</h1>
<div>
  Enter values for your thing in the following form:
</div>
<form method="POST" action="/home/modelbinding?id=3">
  <input name="color" value="Red" />
  <input type="submit" />
</form>
@if (Model is not null)
{
<h2>Submitted Thing</h2>
<hr />
<div>
  <dl class="dl-horizontal">
    <dt>Model.Id</dt>
    <dd>@Model.Id</dd>
    <dt>Model.Color</dt>
    <dd>@Model.Color</dd>
  </dl>
</div>
}
```

1. In `Views/Home`, open `Index.cshtml`, and in the first `<div>`, after rendering the current time, add a new paragraph with a link to the model binding page, as shown in the following markup:

```
<p><a asp-action="ModelBinding" asp-controller="Home">Binding</a></p>
```

1. Start the website.
2. On the home page, click **Binding**.

3. Note the unhandled exception about an ambiguous match, as shown in *Figure 14.6*:

Figure 14.6: An unhandled ambiguous action method mismatch exception

4. Close Chrome and shut down the web server.

Disambiguating action methods

Although the C# compiler can differentiate between the two methods by noting that the signatures are different, from the routing of an HTTP request's point of view, both methods are potential matches. We need an HTTP-specific way to disambiguate the action methods.

We could do this by creating different names for the actions or by specifying that one method should be used for a specific HTTP verb, like `GET`, `POST`, or `DELETE`. That is how we will solve the problem:

1. In `HomeController`, decorate the second `ModelBinding` action method to indicate that it should be used for processing HTTP `POST` requests, that is, when a form is submitted, as shown highlighted in the following code:

```
[HttpPost] // use this action method to process POSTs
public IActionResult ModelBinding(Thing thing)
```

The other `ModelBinding` action method will implicitly be used for all other types of HTTP request, like `GET`, `PUT`, `DELETE`, and so on.

1. Start the website.
2. On the home page, click **Binding**.

3. Click the **Submit** button and note the value for the `Id` property is set from the query string parameter and the value for the color property is set from the form parameter, as shown in *Figure 14.7*:

Figure 14.7: The Model Binding Demo page

4. Close Chrome and shut down the web server.

Passing a route parameter

Now we will set the property using a route parameter:

1. Modify the action for the form to pass the value `2` as a route parameter, as shown highlighted in the following markup:

```
<form method="POST" action="/home/modelbinding/2?id=3">
```

1. Start the website.
2. On the home page, click **Binding**.
3. Click the **Submit** button and note the value for the `Id` property is set from the route parameter and the value for the `Color` property is set from the form parameter.
4. Close Chrome and shut down the web server.

Passing a form parameter

Now we will set the property using a form parameter:

1. Modify the action for the form to pass the value 1 as a form parameter, as shown highlighted in the following markup:

```
<form method="POST" action="/home/modelbinding/2?id=3">
  <input name="id" value="1" />
  <input name="color" value="Red" />
```

```
  <input type="submit" />
</form>
```

1. Start the website.
2. On the home page, click **Binding**.
3. Click the **Submit** button and note the values for the `Id` and `Color` properties are both set from the form parameters.

Good Practice: If you have multiple parameters with the same name, then remember that form parameters have the highest priority and query string parameters have the lowest priority for automatic model binding.

Validating the model

The process of model binding can cause errors, for example, data type conversions or validation errors if the model has been decorated with validation rules. What data has been bound and any binding or validation errors are stored in `ControllerBase.ModelState`.

Let's explore what we can do with model state by applying some validation rules to the bound model and then showing invalid data messages in the view:

1. In the `Models` folder, open `Thing.cs`.
2. Import the namespace for working with validation attributes, as shown in the following code:

```
using System.ComponentModel.DataAnnotations;
```

1. Decorate the `Id` property with a validation attribute to limit the range of allowed numbers to 1 to 10, and one to ensure that the visitor supplies a color, and add a new `Email` property with a regular expression for validation, as shown highlighted in the following code:

```
public record Thing(
  [Range(1, 10)] int? Id,
  [Required] string? Color,
  [EmailAddress] string? Email
);
```

1. In the `Models` folder, add a new class file named `HomeModelBindingViewModel.cs`.
2. Modify its contents to define a record with properties to store the bound model, a flag to indicate that there are errors, and a sequence of error messages, as shown in the following code:

```
namespace Northwind.Mvc.Models;
public record HomeModelBindingViewModel(Thing Thing, bool HasErrors,
  IEnumerable<string> ValidationErrors);
```

1. In `HomeController`, in the `ModelBinding` method that handles HTTP `POST`, delete the previous statement that passed the thing to the view, and instead add statements to create an instance of the view model. Validate the model and store an array of error messages, and then pass the view model to the view, as shown in the following code:

```
HomeModelBindingViewModel model = new(
  Thing: thing, HasErrors: !ModelState.IsValid,
  ValidationErrors: ModelState.Values
    .SelectMany(state => state.Errors)
    .Select(error => error.ErrorMessage)
);
return View(model);
```

1. In `Views\Home`, open `ModelBinding.cshtml`.
2. Modify the model type declaration to use the view model class, as shown in the following markup:

```
@model Northwind.Mvc.Models.HomeModelBindingViewModel
```

1. Add a `<div>` to show any model validation errors, and change the output of the thing's properties because the view model has changed, as shown highlighted in the following markup:

```
<form method="POST" action="/home/modelbinding/2?id=3">
  <input name="id" value="1" />
  <input name="color" value="Red" />
  <input name="email" value="test@example.com" />
  <input type="submit" />
</form>
@if (Model is not null)
{
  <h2>Submitted Thing</h2>
  <hr />
  <div>
    <dl class="dl-horizontal">
      <dt>Model.Thing.Id</dt>
      <dd>@Model.Thing.Id</dd>
      <dt>Model.Thing.Color</dt>
      <dd>@Model.Thing.Color</dd>
      <dt>Model.Thing.Email</dt>
      <dd>@Model.Thing.Email</dd>
    </dl>
  </div>
  @if (Model.HasErrors)
  {
    <div>
      @foreach(string errorMessage in Model.ValidationErrors)
      {
        <div class="alert alert-danger" role="alert">@errorMessage</div>
      }
    </div>
  }
}
```

1. Start the website.
2. On the home page, click **Binding**.
3. Click the **Submit** button and note that `1`, `Red`, and `test@example.com` are valid values.

4. Enter an `Id` of `13`, clear the color textbox, delete the `@` from the email address, click the **Submit** button, and note the error messages, as shown in *Figure 14.8*:

Figure 14.8: The Model Binding Demo page with field validations

5. Close Chrome and shut down the web server.

Good Practice: What regular expression does Microsoft use for the implementation of the `EmailAddress` validation attribute? Find out at the following link: https://github.com/microsoft/referencesource/blob/5697c29004a34d80acdaf5742d7e699022c64ecd/System.ComponentModel.DataAnnotations/DataAnnotations/EmailAddressAttribute.cs#L54

Understanding view helper methods

While creating a view for ASP.NET Core MVC, you can use the `Html` object and its methods to generate markup.

Some useful methods include the following:

- `ActionLink`: Use this to generate an anchor `<a>` element that contains a URL path to the specified controller and action. For example, `Html.ActionLink(linkText: "Binding", actionName: "ModelBinding", controllerName: "Home")` would generate `<a href="/home/modelbinding">Binding</a>`. You can achieve the same result using the anchor tag helper: `<a asp-action="ModelBinding" asp-controller="Home">Binding</a>`.
- `AntiForgeryToken`: Use this inside a `<form>` to insert a `<hidden>` element containing an anti-forgery token that will be validated when the form is submitted.
- `Display` and `DisplayFor`: Use this to generate HTML markup for the expression relative to the current model using a display template. There are built-in display templates for .NET types and custom templates can be created in the `DisplayTemplates` folder. The folder name is case-sensitive on case-sensitive filesystems.
- `DisplayForModel`: Use this to generate HTML markup for an entire model instead of a single expression.
- `Editor` and `EditorFor`: Use this to generate HTML markup for the expression relative to the current model using an editor template. There are built-in editor templates for .NET types that use `<label>` and `<input>`

elements, and custom templates can be created in the `EditorTemplates` folder. The folder name is case-sensitive on case-sensitive filesystems.

- `EditorForModel` : Use this to generate HTML markup for an entire model instead of a single expression.
- `Encode` : Use this to safely encode an object or string into HTML. For example, the string value `"<script>"` would be encoded as `"<script>"` . This is not normally necessary since the Razor `@` symbol encodes string values by default.
- `Raw` : Use this to render a string value *without* encoding as HTML.
- `PartialAsync` and `RenderPartialAsync` : Use these to generate HTML markup for a partial view. You can optionally pass a model and view data.

Querying a database and using display templates

Let's create a new action method that can have a query string parameter passed to it and use that to query the Northwind database for products that cost more than a specified price.

In previous examples, we defined a view model that contained properties for every value that needed to be rendered in the view. In this example, there will be two values: a list of products and the price the visitor entered. To avoid having to define a class or record for the view model, we will pass the list of products as the model and store the maximum price in the `ViewData` collection.

Let's implement this feature:

1. In `HomeController` , import the `Microsoft.EntityFrameworkCore` namespace. We need this to add the `Include` extension method so that we can include related entities, as you learned in *Chapter 10*, *Working with Data Using Entity Framework Core*.
2. Add a new action method, as shown in the following code:

```
public IActionResult ProductsThatCostMoreThan(decimal? price)
{
  if (!price.HasValue)
  {
    return BadRequest("You must pass a product price in the query string, for example, /Home/Proc
  }
  IEnumerable<Product> model = db.Products
    .Include(p => p.Category)
    .Include(p => p.Supplier)
    .Where(p => p.UnitPrice > price);
  if (!model.Any())
  {
    return NotFound(
      $"No products cost more than {price:C}.");
  }
  ViewData["MaxPrice"] = price.Value.ToString("C");
  return View(model); // pass model to view
}
```

1. In the **Views**/**Home** folder, add a new file named `ProductsThatCostMoreThan.cshtml` .
2. Modify the contents, as shown in the following code:

```
@using Packt.Shared
@model IEnumerable<Product>
@{
  string title =
    "Products That Cost More Than " + ViewData["MaxPrice"];
  ViewData["Title"] = title;
}
<h2>@title</h2>
@if (Model is null)
{
  <div>No products found.</div>
}
else
{
```

```
  <table class="table">
    <thead>
      <tr>
        <th>Category Name</th>
        <th>Supplier's Company Name</th>
        <th>Product Name</th>
        <th>Unit Price</th>
        <th>Units In Stock</th>
      </tr>
    </thead>
    <tbody>
    @foreach (Product p in Model)
    {
      <tr>
        <td>
          @Html.DisplayFor(modelItem => p.Category.CategoryName)
        </td>
        <td>
          @Html.DisplayFor(modelItem => p.Supplier.CompanyName)
        </td>
        <td>
          @Html.DisplayFor(modelItem => p.ProductName)
        </td>
        <td>
          @Html.DisplayFor(modelItem => p.UnitPrice)
        </td>
        <td>
          @Html.DisplayFor(modelItem => p.UnitsInStock)
        </td>
      </tr>
    }
    <tbody>
  </table>
}
```

1. In the `Views/Home` folder, open `Index.cshtml`.
2. Add the following form element below the visitor count and above the **Products** heading and its listing of products. This will provide a form for the user to enter a price. The user can then click **Submit** to call the action method that shows only products that cost more than the entered price:

```
<h3>Query products by price</h3>
<form asp-action="ProductsThatCostMoreThan" method="GET">
  <input name="price" placeholder="Enter a product price" />
  <input type="submit" />
</form>
```

1. Start the website.
2. On the home page, enter a price in the form, for example, `50`, and then click on **Submit**.

3. Note the table of the products that cost more than the price that you entered, as shown in *Figure 14.9*:

Figure 14.9: A filtered list of products that cost more than £50

4. Close Chrome and shut down the web server.

Improving scalability using asynchronous tasks

When building a desktop or mobile app, multiple tasks (and their underlying threads) can be used to improve responsiveness, because while one thread is busy with the task, another can handle interactions with the user.

Tasks and their threads can be useful on the server side too, especially with websites that work with files, or request data from a store or a web service that could take a while to respond. But they are detrimental to complex calculations that are CPU-bound, so leave these to be processed synchronously as normal.

When an HTTP request arrives at the web server, a thread from its pool is allocated to handle the request. But if that thread must wait for a resource, then it is blocked from handling any more incoming requests. If a website receives more simultaneous requests than it has threads in its pool, then some of those requests will respond with a server timeout error, **503 Service Unavailable**.

The threads that are locked are not doing useful work. They *could* handle one of those other requests but only if we implement asynchronous code in our websites.

Whenever a thread is waiting for a resource it needs, it can return to the thread pool and handle a different incoming request, improving the scalability of the website, that is, increasing the number of simultaneous requests it can handle.

Why not just have a larger thread pool? In modern operating systems, every thread in the pool has a 1 MB stack. An asynchronous method uses a smaller amount of memory. It also removes the need to create new threads in the pool, which takes time. The rate at which new threads are added to the pool is typically one every two seconds, which is a looooooong time compared to switching between asynchronous threads.

Good Practice: Make your controller action methods asynchronous.

Making controller action methods asynchronous

It is easy to make an existing action method asynchronous:

1. In `HomeController.cs`, modify the `Index` action method to be asynchronous, as shown highlighted in the following code:

```
public async Task<IActionResult> Index()
```

1. In the `Index` action method, await the calls to asynchronous methods to get the categories and products, as shown highlighted in the following code:

```
HomeIndexViewModel model = new
(
  VisitorCount = Random.Shared.Next(1, 1001),
  Categories = await db.Categories.ToListAsync(),
  Products = await db.Products.ToListAsync()
);
```

1. Modify the `ProductDetail` action method in a similar way, as shown highlighted in the following code:

```
public async Task<IActionResult> ProductDetail(int? id)
```

1. In the `ProductDetail` action method, await the calls to asynchronous methods to get the product, as shown highlighted in the following code:

```
Product? model = await db.Products
  .SingleOrDefaultAsync(p => p.ProductId == id);
```

1. Start the website.
2. Note that the functionality of the website is the same, but trust that it will now scale better.
3. Close Chrome and shut down the web server.

Practicing and exploring

Test your knowledge and understanding by answering some questions, get some hands-on practice, and explore this chapter's topics with deeper research.

Exercise 14.1 – Test your knowledge

Answer the following questions:

1. What do the files with the special names `_ViewStart` and `_ViewImports` do when created in the `Views` folder?
2. What are the names of the three segments defined in the default ASP.NET Core MVC route, what do they represent, and which are optional?
3. What does the default model binder do, and what data types can it handle?
4. In a shared layout file like `_Layout.cshtml`, how do you output the content of the current view?
5. In a shared layout file like `_Layout.cshtml`, how do you output a section that the current view can supply content for, and how does the view supply the contents for that section?
6. When calling the `View` method inside a controller's action method, what paths are searched for the view by convention?
7. How can you instruct the visitor's browser to cache the response for 24 hours?
8. Why might you enable Razor Pages even if you are not creating any yourself?
9. How does ASP.NET Core MVC identify classes that can act as controllers?
10. In what ways does ASP.NET Core MVC make it easier to test a website?

Exercise 14.2 – Practice implementing MVC by implementing a category detail page

The `Northwind.Mvc` project has a home page that shows categories, but when the `View` button is clicked, the website returns a `404 Not Found` error, for example, for the following URL:

```
https://localhost:5001/category/1
```

Extend the `Northwind.Mvc` project by adding the ability to show a detail page for a category.

Exercise 14.3 – Practice improving scalability by understanding and implementing async action methods

Almost a decade ago, Stephen Cleary wrote an excellent article for MSDN Magazine explaining the scalability benefits of implementing async action methods for ASP.NET. The same principles apply to ASP.NET Core, but even more so, because unlike the old ASP.NET as described in the article, ASP.NET Core supports asynchronous filters and other components.

Read the article at the following link:

https://docs.microsoft.com/en-us/archive/msdn-magazine/2014/october/async-programming-introduction-to-async-await-on-asp-net

Exercise 14.4 – Practice unit testing MVC controllers

Controllers are where the business logic of your website runs, so it is important to test the correctness of that logic using unit tests, as you learned in *Chapter 4*, *Writing, Debugging, and Testing Functions*.

Write some unit tests for `HomeController`.

> **Good Practice**: You can read more about how to unit test controllers at the following link: https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/testing

Exercise 14.5 – Explore topics

Use the links on the following page to learn more about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-14---building-websites-using-the-model-view-controller-pattern

Summary

In this chapter, you learned how to build large, complex websites in a way that is easy to unit test by registering and injecting dependency services like database contexts and loggers and is easier to manage with teams of programmers using ASP.NET Core MVC. You learned about:

- Configuration.
- Authentication.
- Routes.
- Models.
- Views.
- Controllers.

In the next chapter, you will learn how to build and consume services that use HTTP as the communication layer, aka web services.

15 Building and Consuming Web Services

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about learning how to build web services (aka HTTP or REST services) using the ASP.NET Core Web API and consuming web services using HTTP clients that could be any other type of .NET app, including a website or a mobile or desktop app.

This chapter requires knowledge and skills that you learned in *Chapter 10*, *Working with Data Using Entity Framework Core*, and *Chapters 12* to *14*, about building websites using ASP.NET Core.

In this chapter, we will cover the following topics:

- Building web services using ASP.NET Core Web API
- Documenting and testing web services
- Consuming web services using HTTP clients
- Implementing advanced features for web services
- Building web services using minimal APIs

Building web services using ASP.NET Core Web API

Before we build a modern web service, we need to cover some background to set the context for this chapter.

Understanding web service acronyms

Although HTTP was designed originally to request and respond with HTML and other resources for humans to look at, it is also good for building services.

Roy Fielding stated in his doctoral dissertation, describing the **Representational State Transfer** (**REST**) architectural style, that the HTTP standard would be good for building services because it defines the following:

- URIs to uniquely identify resources, like https://localhost:5001/api/products/23.
- Methods to perform common tasks on those resources, like `GET`, `POST`, `PUT`, and `DELETE`.
- The ability to negotiate the media type of content exchanged in requests and responses, such as XML and JSON. Content negotiation happens when the client specifies a request header like `Accept: application/xml,*/*;q=0.8`. The default response format used by the ASP.NET Core Web API is JSON, which means one of the response headers would be `Content-Type: application/json; charset=utf-8`.

Web services use the HTTP communication standard, so they are sometimes called HTTP or RESTful services. HTTP or RESTful services are what this chapter is about.

Understanding HTTP requests and responses for Web APIs

HTTP defines standard types of requests and standard codes to indicate a type of response. Most of them can be used to implement Web API services.

The most common type of request is `GET`, to retrieve a resource identified by a unique path, with additional options like what media type is acceptable, set as request headers, as shown in the following example:

```
GET /path/to/resource
Accept: application/json
```

Common responses include success and multiple types of failure, as shown in the following table:

Status code	Description
`200 OK`	The path was correctly formed, the resource was successfully found, serialized into an acceptable media type, and then returned in the response body. The response headers specify the Content-Type, Content-Length, and Content-Encoding, for example, GZIP.
`301 Moved Permanently`	Over time a web service may change its resource model including the path used to identify an existing resource. The web service can indicate the new path by returning this status code and a response header named Location that has the new path.
`302 Found`	Similar to 301.
`304 Not Modified`	If the request included the If-Modified-Since header, then the web service can respond with this status code. The response body is empty because the client should use its cached copy of the resource.
`400 Bad Request`	The request was invalid, for example, it used a path for a product using an integer ID where the ID value is missing.
`401 Unauthorized`	The request was valid, the resource was found, but the client did not supply credentials or is not authorized to access that resource. Re-authenticating may enable access, for example, by adding or changing the Authorization request header.
`403 Forbidden`	The request was valid, the resource was found, but the client is not authorized to access that resource. Re-authenticating will not fix the issue.
`404 Not Found`	The request was valid, but the resource was not found. The resource may be found if the request is repeated later. To indicate that a resource will never be found, return 410 Gone.
`406 Not Acceptable`	If the request has an Accept header that only lists media types that the web service does not support. For example, if the client requests JSON but the web service can only return XML.
`451 Unavailable for Legal Reasons`	A website hosted in the USA might return this for requests coming from Europe to avoid having to comply with the General Data Protection Regulation (GDPR). The number was chosen as a reference to the novel Fahrenheit 451 in which books are banned and burned.
`500 Server Error`	The request was valid, but something went wrong on the server side while processing the request. Retrying again later might work.
`503 Service Unavailable`	The web service is busy and cannot handle the request. Trying again later might work.

Other common types of HTTP requests include `POST`, `PUT`, `PATCH`, or `DELETE` that create, modify, or delete resources.

To create a new resource, you might make a `POST` request with a body that contains the new resource, as shown in the following code:

```
POST /path/to/resource
Content-Length: 123
Content-Type: application/json
```

To create a new resource or update an existing resource, you might make a `PUT` request with a body that contains a whole new version of the existing resource, and if the resource does not exist, it is created, or if it does exist, it is replaced (sometimes called an **upsert** operation), as shown in the following code:

```
PUT /path/to/resource
Content-Length: 123
Content-Type: application/json
```

To update an existing resource more efficiently, you might make a `PATCH` request with a body that contains an object with only the properties that need changing, as shown in the following code:

```
PATCH /path/to/resource
Content-Length: 123
Content-Type: application/json
```

To delete an existing resource, you might make a `DELETE` request, as shown in the following code:

```
DELETE /path/to/resource
```

As well as the responses shown in the table above for a `GET` request, all the types of requests that create, modify, or delete a resource have additional possible common responses, as shown in the following table:

Status code	Description
`201 Created`	The new resource was created successfully, the response header named Location contains its path, and the response body contains the newly created resource. Immediately GET-ing the resource should return 200.
`202 Accepted`	The new resource cannot be created immediately so the request is queued for later processing and immediately GET-ing the resource might return 404. The body can contain a resource that points to some form of status checker or an estimate of when the resource will become available.
`204 No Content`	Commonly used in response to a DELETE request since returning the resource in the body after deleting it does not usually make sense! Sometimes used in response to POST, PUT, or PATCH requests if the client does not need to confirm that the request was processed correctly.
`405 Method Not Allowed`	Returned when the request used a method that is not supported. For example, a web service designed to be read-only may explicitly disallow PUT, DELETE, and so on.
`415 Unsupported Media Type`	Returned when the resource in the request body uses a media type that the web service cannot handle. For example, if the body contains a resource in XML format but the web service can only process JSON.

Creating an ASP.NET Core Web API project

We will build a web service that provides a way to work with data in the Northwind database using ASP.NET Core so that the data can be used by any client application on any platform that can make HTTP requests and receive HTTP responses:

1. Use your preferred code editor to open the `PracticalApps` solution/workspace and then add a new project, as defined in the following list:
   I. Project template: **ASP.NET Core Web API** / `webapi`
   II. Workspace/solution file and folder: `PracticalApps`
   III. Project file and folder: `Northwind.WebApi`
   IV. Other Visual Studio options: **Authentication Type**: None, **Configure for HTTPS**: selected, **Enable Docker**: cleared, **Use controllers (uncheck to use minimal APIs)**: selected, **Enable OpenAPI**

support: selected.

2. In Visual Studio Code, select `Northwind.WebApi` as the active OmniSharp project.
3. Build the `Northwind.WebApi` project.
4. In the `Controllers` folder, open and review `WeatherForecastController.cs`, as shown in the following code:

```
using Microsoft.AspNetCore.Mvc;
namespace Northwind.WebApi.Controllers;
[ApiController]
[Route("[controller]")]
public class WeatherForecastController : ControllerBase
{
  private static readonly string[] Summaries = new[]
  {
    "Freezing", "Bracing", "Chilly", "Cool", "Mild",
    "Warm", "Balmy", "Hot", "Sweltering", "Scorching"
  };
  private readonly ILogger<WeatherForecastController> _logger;
  public WeatherForecastController(
    ILogger<WeatherForecastController> logger)
  {
    _logger = logger;
  }
  [HttpGet(Name = "GetWeatherForecast")]
  public IEnumerable<WeatherForecast> Get()
  {
    return Enumerable.Range(1, 5).Select(index => new WeatherForecast
      {
        Date = DateTime.Now.AddDays(index),
        TemperatureC = Random.Shared.Next(-20, 55),
        Summary = Summaries[Random.Shared.Next(Summaries.Length)]
      })
      .ToArray();
  }
}
```

While reviewing the preceding code, note the following:

1. The `Controller` class inherits from `ControllerBase`. This is simpler than the `Controller` class used in MVC because it does not have methods like `View` to generate HTML responses by passing a view model to a Razor file.
2. The `[Route]` attribute registers the `/weatherforecast` relative URL for clients to use to make HTTP requests that will be handled by this controller. For example, an HTTP request for `https://localhost:5001/weatherforecast/` would be handled by this controller. Some developers like to prefix the controller name with `api/`, which is a convention to differentiate between MVC and Web API in mixed projects. If you use `[controller]` as shown, it uses the characters before `Controller` in the class name, in this case, `WeatherForecast`, or you can simply enter a different name without the square brackets, for example, `[Route("api/forecast")]`.
3. The `[ApiController]` attribute was introduced with ASP.NET Core 2.1 and it enables REST-specific behavior for controllers, like automatic HTTP `400` responses for invalid models, as you will see later in this chapter.
4. The `[HttpGet]` attribute registers the `Get` method in the `Controller` class to respond to HTTP `GET` requests, and its implementation uses the shared `Random` object to return an array of `WeatherForecast` objects with random temperatures and summaries like `Bracing` or `Balmy` for the next five days of weather.

1. In `WeatherForecastController.cs`, add a second `Get` method that allows the call to specify how many days ahead the forecast should be by implementing the following:
   I. Add a comment above the original method to show the action method and URL path that it responds to.
   II. Add a new method with an integer parameter named `days`.
   III. Cut and paste the original `Get` method implementation code statements into the new `Get` method.
   IV. Modify the new method to create an `IEnumerable` of integers up to the number of days requested, and modify the original `Get` method to call the new `Get` method and pass the value `5`.

Your methods should be as shown in the following code:

```
// GET /weatherforecast
[HttpGet(Name = "GetWeatherForecastFiveDays")]
public IEnumerable<WeatherForecast> Get() // original method
{
  return Get(days: 5); // five day forecast
}
// GET /weatherforecast/7
[HttpGet(template: "{days:int}", Name = "GetWeatherForecast")]
public IEnumerable<WeatherForecast> Get(int days) // new method
{
  return Enumerable.Range(1, days).Select(index => new WeatherForecast
    {
      Date = DateTime.Now.AddDays(index),
      TemperatureC = Random.Shared.Next(-20, 55),
      Summary = Summaries[Random.Shared.Next(Summaries.Length)]
    })
    .ToArray();
}
```

In the `[HttpGet]` attribute, note the route template pattern `{days:int}` that constrains the `days` parameter to `int` values.

Reviewing the web service's functionality

Now, we will test the web service's functionality:

1. If you are using Visual Studio, in **Properties**, open the `launchSettings.json` file, and note that by default, it will launch the browser and navigate to the `/swagger` relative URL path, as shown highlighted in the following markup:

```
"profiles": {
  "Northwind.WebApi": {
    "commandName": "Project",
    "dotnetRunMessages": "true",
    "launchBrowser": true,
    "launchUrl": "swagger",
    "applicationUrl": "https://localhost:5001;http://localhost:5000",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
```

1. Modify the profile named `Northwind.WebApi` to set `launchBrowser` to `false`.
2. For the `applicationUrl`, change the random port number for `HTTP` to `5000` and for `HTTPS` to `5001`.
3. The solution on GitHub is configured to use port `5002` because we will change its configuration later in the book.
4. Start the web service project.
5. Start Chrome.
6. Navigate to `https://localhost:5001/` and note you will get a `404` status code response because we have not enabled static files and there is not an `index.xhtml`, nor is there an MVC controller with a route configured, either. Remember that this project is not designed for a human to view and interact with, so this is expected behavior for a web service.
7. In Chrome, show **Developer tools**.

8. Navigate to https://localhost:5001/weatherforecast and note the Web API service should return a JSON document with five random weather forecast objects in an array, as shown in *Figure 15.1*:

Figure 15.1: A request and response from a weather forecast web service

9. Close **Developer tools**.
10. Navigate to https://localhost:5001/weatherforecast/14 and note the response when requesting a two-week weather forecast contains 14 forecasts.
11. Close Chrome and shut down the web server.

Creating a web service for the Northwind database

Unlike MVC controllers, Web API controllers do not call Razor views to return HTML responses for website visitors to see in browsers. Instead, they use **content negotiation** with the client application that made the HTTP request to return data in formats such as XML, JSON, or X-WWW-FORM-URLENCODED in their HTTP response.

The client application must then deserialize the data from the negotiated format. The most used format for modern web services is **JavaScript Object Notation** (**JSON**) because it is compact and works natively with JavaScript in

a browser when building **Single-Page Applications** (**SPAs**) with client-side technologies like Angular, React, and Vue.

We will reference the Entity Framework Core entity data model for the Northwind database that you created in *Chapter 12*, *Introducing Web Development Using ASP.NET Core*:

1. In the `Northwind.WebApi` project, add a project reference to `Northwind.Common.DataContext` for either SQLite or SQL Server, as shown in the following markup:

```
<ItemGroup>
  <!-- change Sqlite to SqlServer if you prefer -->
  <ProjectReference Include=
"..\Northwind.Common.DataContext.Sqlite\Northwind.Common.DataContext.Sqlite.csproj" />
</ItemGroup>
```

1. In the `Northwind.WebApi` project, globally and statically import the `System.Console` class.
2. Build the `Northwind.WebApi` project and fix any compile errors in your code.
3. In `Program.cs`, import namespaces for working with web media formatters and the shared Packt classes, as shown in the following code:

```
using Microsoft.AspNetCore.Mvc.Formatters; // IOutputFormatter, OutputFormatter
using Packt.Shared; // AddNorthwindContext extension method
```

1. In `Program.cs`, add a statement before the call to `AddControllers` to register the `Northwind` database context class (it will use either SQLite or SQL Server depending on which database provider you referenced in the project file), as shown in the following code:

```
builder.Services.AddNorthwindContext();
```

1. In the call to `AddControllers`, add a lambda block with statements to write the names and supported media types of the default output formatters to the console, and then add XML serializer formatters, as shown in the following code:

```
builder.Services.AddControllers(options =>
{
  WriteLine("Default output formatters:");
  foreach (IOutputFormatter formatter in options.OutputFormatters)
  {
    OutputFormatter? mediaFormatter = formatter as OutputFormatter;
    if (mediaFormatter is null)
    {
      WriteLine($"  {formatter.GetType().Name}");
    }
    else // OutputFormatter class has SupportedMediaTypes
    {
      WriteLine("  {0}, Media types: {1}",
        arg0: mediaFormatter.GetType().Name,
        arg1: string.Join(", ",
          mediaFormatter.SupportedMediaTypes));
    }
  }
})
.AddXmlDataContractSerializerFormatters()
.AddXmlSerializerFormatters();
```

1. Start the web service.
2. In a command prompt or terminal, note that there are four default output formatters, including ones that convert `null` values into `204 No Content` and ones to support responses that are plain text, byte streams, and JSON, as shown in the following output:

```
Default output formatters:
  HttpNoContentOutputFormatter
  StringOutputFormatter, Media types: text/plain
  StreamOutputFormatter
  SystemTextJsonOutputFormatter, Media types: application/json, text/json, application/*+json
```

1. Shut down the web server.

Creating data repositories for entities

Defining and implementing a data repository to provide CRUD operations is good practice. The CRUD acronym includes the following operations:

- C for Create
- R for Retrieve (or Read)
- U for Update
- D for Delete

We will create a data repository for the `Customers` table in Northwind. There are only 91 customers in this table, so we will store a copy of the whole table in memory to improve scalability and performance when reading customer records.

> **Good Practice**: In a real web service, you should use a distributed cache like Redis, an open-source data structure store that can be used as a high-performance, high-availability database, cache, or message broker.

We will follow modern good practice and make the repository API asynchronous. It will be instantiated by a `Controller` class using constructor parameter injection, so a new instance is created to handle every HTTP request:

1. In the `Northwind.WebApi` project, create a folder named `Repositories`.
2. Add a new interface file and a class file to the `Repositories` folder named `ICustomerRepository.cs` and `CustomerRepository.cs`.
3. In `ICustomerRepository.cs`, define an interface with five CRUD methods, as shown in the following code:

```
using Packt.Shared; // Customer
namespace Northwind.WebApi.Repositories;
public interface ICustomerRepository
{
  Task<Customer?> CreateAsync(Customer c);
  Task<IEnumerable<Customer>> RetrieveAllAsync();
  Task<Customer?> RetrieveAsync(string id);
  Task<Customer?> UpdateAsync(string id, Customer c);
  Task<bool?> DeleteAsync(string id);
}
```

1. In `CustomerRepository.cs`, define a class with five implemented methods, remembering that methods that use `await` inside them must be marked as `async`, as shown in the following code:

```
using Microsoft.EntityFrameworkCore.ChangeTracking; // EntityEntry<T>
using Packt.Shared; // Customer
using System.Collections.Concurrent; // ConcurrentDictionary
namespace Northwind.WebApi.Repositories;
public class CustomerRepository : ICustomerRepository
{
  // use a static thread-safe dictionary field to cache the customers
  private static ConcurrentDictionary
    <string, Customer>? customersCache;
  // use an instance data context field because it should not be
  // cached due to their internal caching
  private NorthwindContext db;
  public CustomerRepository(NorthwindContext injectedContext)
  {
    db = injectedContext;
    // pre-load customers from database as a normal
    // Dictionary with CustomerId as the key,
    // then convert to a thread-safe ConcurrentDictionary
    if (customersCache is null)
    {
      customersCache = new ConcurrentDictionary<string, Customer>(
        db.Customers.ToDictionary(c => c.CustomerId));
```

```
  }
}
public async Task<Customer?> CreateAsync(Customer c)
{
  // normalize CustomerId into uppercase
  c.CustomerId = c.CustomerId.ToUpper();
  // add to database using EF Core
  EntityEntry<Customer> added = await db.Customers.AddAsync(c);
  int affected = await db.SaveChangesAsync();
  if (affected == 1)
  {
    if (customersCache is null) return c;
    // if the customer is new, add it to cache, else
    // call UpdateCache method
    return customersCache.AddOrUpdate(c.CustomerId, c, UpdateCache);
  }
  else
  {
    return null;
  }
}
public Task<IEnumerable<Customer>> RetrieveAllAsync()
{
  // for performance, get from cache
  return Task.FromResult(customersCache is null
      ? Enumerable.Empty<Customer>() : customersCache.Values);
}
public Task<Customer?> RetrieveAsync(string id)
{
  // for performance, get from cache
  id = id.ToUpper();
  if (customersCache is null) return null!;
  customersCache.TryGetValue(id, out Customer? c);
  return Task.FromResult(c);
}
private Customer UpdateCache(string id, Customer c)
{
  Customer? old;
  if (customersCache is not null)
  {
    if (customersCache.TryGetValue(id, out old))
    {
      if (customersCache.TryUpdate(id, c, old))
      {
        return c;
      }
    }
  }
  return null!;
}
public async Task<Customer?> UpdateAsync(string id, Customer c)
{
  // normalize customer Id
  id = id.ToUpper();
  c.CustomerId = c.CustomerId.ToUpper();
  // update in database
  db.Customers.Update(c);
  int affected = await db.SaveChangesAsync();
  if (affected == 1)
  {
    // update in cache
    return UpdateCache(id, c);
  }
  return null;
}
public async Task<bool?> DeleteAsync(string id)
{
  id = id.ToUpper();
  // remove from database
  Customer? c = db.Customers.Find(id);
  if (c is null) return null;
  db.Customers.Remove(c);
```

```
    int affected = await db.SaveChangesAsync();
    if (affected == 1)
    {
      if (customersCache is null) return null;
      // remove from cache
      return customersCache.TryRemove(id, out c);
    }
    else
    {
      return null;
    }
  }
}
```

Implementing a Web API controller

There are some useful attributes and methods for implementing a controller that returns data instead of HTML.

With MVC controllers, a route like `/home/index` tells us the controller class name and the action method name, for example, the `HomeController` class and the `Index` action method.

With Web API controllers, a route like `/weatherforecast` only tells us the controller class name, for example, `WeatherForecastController`. To determine the action method name to execute, we must map HTTP methods like `GET` and `POST` to methods in the controller class.

You should decorate controller methods with the following attributes to indicate the HTTP method that they will respond to:

- `[HttpGet]`, `[HttpHead]`: These action methods respond to `GET` or `HEAD` requests to retrieve a resource and return either the resource and its response headers or just the response headers.
- `[HttpPost]`: This action method responds to `POST` requests to create a new resource or perform some other action defined by the service.
- `[HttpPut]`, `[HttpPatch]`: These action methods respond to `PUT` or `PATCH` requests to update an existing resource either by replacing it or updating a subset of its properties.
- `[HttpDelete]`: This action method responds to `DELETE` requests to remove a resource.
- `[HttpOptions]`: This action method responds to `OPTIONS` requests.

Understanding action method return types

An action method can return .NET types like a single `string` value, complex objects defined by a `class`, `record`, or `struct`, or collections of complex objects. The ASP.NET Core Web API will serialize them into the requested data format set in the HTTP request `Accept` header, for example, JSON, if a suitable serializer has been registered.

For more control over the response, there are helper methods that return an `ActionResult` wrapper around the .NET type.

Declare the action method's return type to be `IActionResult` if it could return different return types based on inputs or other variables. Declare the action method's return type to be `ActionResult<T>` if it will only return a single type but with different status codes.

> **Good Practice**: Decorate action methods with the `[ProducesResponseType]` attribute to indicate all the known types and HTTP status codes that the client should expect in a response. This information can then be publicly exposed to document how a client should interact with your web service. Think of it as part of your formal documentation. Later in this chapter, you will learn how you can install a code analyzer to give you warnings when you do not decorate your action methods like this.

For example, an action method that gets a product based on an id parameter would be decorated with three attributes – one to indicate that it responds to `GET` requests and has an id parameter, and two to indicate what

happens when it succeeds and when the client has supplied an invalid product ID, as shown in the following code:

```
[HttpGet("{id}")]
[ProducesResponseType(200, Type = typeof(Product))]
[ProducesResponseType(404)]
public IActionResult Get(string id)
```

The `ControllerBase` class has methods to make it easy to return different responses, as shown in the following table:

Method	Description
`Ok`	Returns a 200 status code and a resource converted to the client's preferred format, like JSON or XML. Commonly used in response to a GET request.
`CreatedAtRoute`	Returns a 201 status code and the path to the new resource. Commonly used in response to a POST request to create a resource that can be performed quickly.
`Accepted`	Returns a 202 status code to indicate the request is being processed but has not completed. Commonly used in response to a POST, PUT, PATCH, or DELETE request that triggers a background process that takes a long time to complete.
`NoContentResult`	Returns a 204 status code and an empty response body. Commonly used in response to a PUT, PATCH, or DELETE request when the response does not need to contain the affected resource.
`BadRequest`	Returns a 400 status code and an optional message string with more details.
`NotFound`	Returns a 404 status code and an automatically populated ProblemDetails body (requires a compatibility version of 2.2 or later).

Configuring the customer repository and Web API controller

Now you will configure the repository so that it can be called from within a Web API controller.

You will register a scoped dependency service implementation for the repository when the web service starts up and then use constructor parameter injection to get it in a new Web API controller for working with customers.

To show an example of differentiating between MVC and Web API controllers using routes, we will use the common `/api` URL prefix convention for the customers controller:

1. In `Program.cs`, import the namespace for working with our customers repository, as shown in the following code:

```
using Northwind.WebApi.Repositories; // ICustomerRepository, CustomerRepository
```

1. In `Program.cs`, add a statement before the call to the `Build` method, which will register the `CustomerRepository` for use at runtime as a scoped dependency, as shown highlighted in the following code:

```
builder.Services.AddScoped<ICustomerRepository, CustomerRepository>();
```

- **Good Practice**: Our repository uses a database context that is registered as a scoped dependency. You can only use scoped dependencies inside other scoped dependencies, so we cannot register the repository as a singleton. You can read more about this at the following link: https://docs.microsoft.com/en-us/dotnet/core/extensions/dependency-injection#scoped

1. In the `Controllers` folder, add a new class named `CustomersController.cs`.
2. In `CustomersController.cs`, add statements to define a Web API controller class to work with customers, as shown in the following code:

```
using Microsoft.AspNetCore.Mvc; // [Route], [ApiController], ControllerBase
using Packt.Shared; // Customer
using Northwind.WebApi.Repositories; // ICustomerRepository
```

```
namespace Northwind.WebApi.Controllers;
// base address: api/customers
[Route("api/[controller]")]
[ApiController]
public class CustomersController : ControllerBase
{
  private readonly ICustomerRepository repo;
  // constructor injects repository registered in Startup
  public CustomersController(ICustomerRepository repo)
  {
    this.repo = repo;
  }
  // GET: api/customers
  // GET: api/customers/?country=[country]
  // this will always return a list of customers (but it might be empty)
  [HttpGet]
  [ProducesResponseType(200, Type = typeof(IEnumerable<Customer>))]
  public async Task<IEnumerable<Customer>> GetCustomers(string? country)
  {
    if (string.IsNullOrWhiteSpace(country))
    {
      return await repo.RetrieveAllAsync();
    }
    else
    {
      return (await repo.RetrieveAllAsync())
        .Where(customer => customer.Country == country);
    }
  }
  // GET: api/customers/[id]
  [HttpGet("{id}", Name = nameof(GetCustomer))] // named route
  [ProducesResponseType(200, Type = typeof(Customer))]
  [ProducesResponseType(404)]
  public async Task<IActionResult> GetCustomer(string id)
  {
    Customer? c = await repo.RetrieveAsync(id);
    if (c == null)
    {
      return NotFound(); // 404 Resource not found
    }
    return Ok(c); // 200 OK with customer in body
  }
  // POST: api/customers
  // BODY: Customer (JSON, XML)
  [HttpPost]
  [ProducesResponseType(201, Type = typeof(Customer))]
  [ProducesResponseType(400)]
  public async Task<IActionResult> Create([FromBody] Customer c)
  {
    if (c == null)
    {
      return BadRequest(); // 400 Bad request
    }
    Customer? addedCustomer = await repo.CreateAsync(c);
    if (addedCustomer == null)
    {
      return BadRequest("Repository failed to create customer.");
    }
    else
    {
      return CreatedAtRoute( // 201 Created
        routeName: nameof(GetCustomer),
        routeValues: new { id = addedCustomer.CustomerId.ToLower() },
        value: addedCustomer);
    }
  }
  // PUT: api/customers/[id]
  // BODY: Customer (JSON, XML)
  [HttpPut("{id}")]
  [ProducesResponseType(204)]
  [ProducesResponseType(400)]
  [ProducesResponseType(404)]
```

```
  public async Task<IActionResult> Update(
    string id, [FromBody] Customer c)
  {
    id = id.ToUpper();
    c.CustomerId = c.CustomerId.ToUpper();
    if (c == null || c.CustomerId != id)
    {
      return BadRequest(); // 400 Bad request
    }
    Customer? existing = await repo.RetrieveAsync(id);
    if (existing == null)
    {
      return NotFound(); // 404 Resource not found
    }
    await repo.UpdateAsync(id, c);
    return new NoContentResult(); // 204 No content
  }
  // DELETE: api/customers/[id]
  [HttpDelete("{id}")]
  [ProducesResponseType(204)]
  [ProducesResponseType(400)]
  [ProducesResponseType(404)]
  public async Task<IActionResult> Delete(string id)
  {
    Customer? existing = await repo.RetrieveAsync(id);
    if (existing == null)
    {
      return NotFound(); // 404 Resource not found
    }
    bool? deleted = await repo.DeleteAsync(id);
    if (deleted.HasValue && deleted.Value) // short circuit AND
    {
      return new NoContentResult(); // 204 No content
    }
    else
    {
      return BadRequest( // 400 Bad request
        $"Customer {id} was found but failed to delete.");
    }
  }
}
```

While reviewing this Web API controller class, note the following:

- The `Controller` class registers a route that starts with `api/` and includes the name of the controller, that is, `api/customers`.
- The constructor uses dependency injection to get the registered repository for working with customers.
- There are five action methods to perform CRUD operations on customers—two `GET` methods (for all customers or one customer), `POST` (create), `PUT` (update), and `DELETE`.
- The `GetCustomers` method can have a `string` parameter passed with a country name. If it is missing, all customers are returned. If it is present, it is used to filter customers by country.
- The `GetCustomer` method has a route explicitly named `GetCustomer` so that it can be used to generate a URL after inserting a new customer.
- The `Create` and `Update` methods both decorate the `customer` parameter with `[FromBody]` to tell the model binder to populate it with values from the body of the `POST` request.
- The `Create` method returns a response that uses the `GetCustomer` route so that the client knows how to get the newly created resource in the future. We are matching up two methods to create and then get a customer.
- The `Create` and `Update` methods do not need to check the model state of the customer passed in the body of the HTTP request and return a `400 Bad Request` containing details of the model validation errors if it is not valid because the controller is decorated with `[ApiController]`, which does this for you.

When an HTTP request is received by the service, then it will create an instance of the `Controller` class, call the appropriate action method, return the response in the format preferred by the client, and release the resources used by the controller, including the repository and its data context.

Specifying problem details

A feature added in ASP.NET Core 2.1 and later is an implementation of a web standard for specifying problem details.

In Web API controllers decorated with `[ApiController]` in a project with ASP.NET Core 2.2 or later compatibility enabled, action methods that return `IActionResult` and return a client error status code, that is, `4xx`, will automatically include a serialized instance of the `ProblemDetails` class in the response body.

If you want to take control, then you can create a `ProblemDetails` instance yourself and include additional information.

Let's simulate a bad request that needs custom data returned to the client:

1. At the top of the implementation of the `Delete` action method, add statements to check if the `id` matches the literal string value `"bad"`, and if so, then return a custom problem details object, as shown in the following code:

```
// take control of problem details
if (id == "bad")
{
  ProblemDetails problemDetails = new()
  {
    Status = StatusCodes.Status400BadRequest,
    Type = "https://localhost:5001/customers/failed-to-delete",
    Title = $"Customer ID {id} found but failed to delete.",
    Detail = "More details like Company Name, Country and so on.",
    Instance = HttpContext.Request.Path
  };
  return BadRequest(problemDetails); // 400 Bad Request
}
```

1. You will test this functionality later.

Controlling XML serialization

In `Program.cs`, we added the `XmlSerializer` so that our Web API service can return XML as well as JSON if the client requests that.

However, the `XmlSerializer` cannot serialize interfaces, and our entity classes use `ICollection<T>` to define related child entities. This causes a warning at runtime, for example, for the `Customer` class and its `Orders` property, as shown in the following output:

```
warn: Microsoft.AspNetCore.Mvc.Formatters.XmlSerializerOutputFormatter[1]
An error occurred while trying to create an XmlSerializer for the type 'Packt.Shared.Customer'.
System.InvalidOperationException: There was an error reflecting type 'Packt.Shared.Customer'.
---> System.InvalidOperationException: Cannot serialize member 'Packt.
Shared.Customer.Orders' of type 'System.Collections.Generic.ICollection`1[[Packt. Shared.Order, N
```

We can prevent this warning by excluding the `Orders` property when serializing a `Customer` to XML:

1. In the `Northwind.Common.EntityModels.Sqlite` and the `Northwind.Common.EntityModels.SqlServer` projects, open `Customer.cs`.
2. Import the namespace so that we can use the `[XmlIgnore]` attribute, as shown highlighted in the following code:

```
using System.Xml.Serialization; // [XmlIgnore]
```

1. Decorate the `Orders` property with an attribute to ignore it when serializing, as shown highlighted in the following code:

```
[InverseProperty(nameof(Order.Customer))]
[XmlIgnore]
public virtual ICollection<Order> Orders { get; set; }
```

1. If you are using SQL Server, in the `Northwind.Common.EntityModels.SqlServer` project, in `Customer.cs`, decorate the `CustomerTypes` property with `[XmlIgnore]` too.

Documenting and testing web services

You can easily test a web service by making HTTP `GET` requests using a browser. To test other HTTP methods, we need a more advanced tool.

Testing GET requests using a browser

You will use Chrome to test the three implementations of a `GET` request – for all customers, for customers in a specified country, and for a single customer using their unique customer ID:

1. Start the `Northwind.WebApi` web service project.
2. Start Chrome.

3. Navigate to `https://localhost:5001/api/customers` and note the JSON document returned, containing all 91 customers in the Northwind database (unsorted), as shown in *Figure 15.2*:

Figure 15.2: Customers from the Northwind database as a JSON document

4. Navigate to https://localhost:5001/api/customers/?country=Germany and note the JSON document returned, containing only the customers in Germany.
5. If you get an empty array returned, then make sure you have entered the country name using the correct casing because the database query is case-sensitive. For example, compare the results of `uk` and `UK`.
6. Navigate to https://localhost:5001/api/customers/alfki and note the JSON document returned containing only the customer named **Alfreds Futterkiste**.

Unlike with country names, we do not need to worry about casing for the customer `id` value because inside the controller class, we normalized the `string` value to uppercase in code.

But how can we test the other HTTP methods, such as `POST`, `PUT`, and `DELETE`? And how can we document our web service so it's easy for anyone to understand how to interact with it?

To solve the first problem, we can install a Visual Studio Code extension named **REST Client**. To solve the second, we can use **Swagger**, the world's most popular technology for documenting and testing HTTP APIs. But first, let's see what is possible with the Visual Studio Code extension.

There are many tools for testing Web APIs, for example, **Postman**. Although Postman is popular, I prefer **REST Client** because it does not hide what is happening. I feel Postman is too GUI-y. But I encourage you to explore different tools and find the ones that fit your style. You can learn more about Postman at the following link:https://www.postman.com/

Testing HTTP requests with the REST Client extension

REST Client is an extension that allows you to send any type of HTTP request and view the response in Visual Studio Code. Even if you prefer to use Visual Studio as your code editor, it is useful to install Visual Studio Code to use an extension like REST Client.

Making GET requests using REST Client

We will start by creating a file for testing `GET` requests:

1. If you have not already installed REST Client by Huachao Mao (`humao.rest-client`), then install it in Visual Studio Code now.
2. In your preferred code editor, open the `PracticalApps` solution/workspace and then start the `Northwind.WebApi` project web service.
3. In Visual Studio Code, in the `PracticalApps` folder, create a `RestClientTests` folder, and then open the folder.
4. In the `RestClientTests` folder, create a file named `get-customers.http` , and modify its contents to contain an HTTP `GET` request to retrieve all customers, as shown in the following code:

```
GET https://localhost:5001/api/customers/ HTTP/1.1
```

1. In Visual Studio Code, navigate to **View** | **Command Palette**, enter `rest client` , select the command **Rest Client: Send Request**, and press Enter, as shown in *Figure 15.3*:

Figure 15.3: Sending an HTTP GET request using REST Client

2. Note the **Response** is shown in a new tabbed window pane vertically and that you can rearrange the open tabs to a horizontal layout by dragging and dropping tabs.
3. In `get-customers.http` , add more `GET` requests, each separated by three hash symbols, to test getting customers in various countries and getting a single customer using their ID, as shown in the following code:

```
###
GET https://localhost:5001/api/customers/?country=Germany HTTP/1.1
###
GET https://localhost:5001/api/customers/?country=USA HTTP/1.1
Accept: application/xml
###
GET https://localhost:5001/api/customers/ALFKI HTTP/1.1
###
GET https://localhost:5001/api/customers/abcxy HTTP/1.1
```

1. Click the **Send Request** link above each request to send it; for example, the `GET` that has a request header to request customers in the USA as XML instead of JSON, as shown in *Figure 15.4*:

Figure 15.4: Sending a request for XML and getting a response using REST Client

Making other requests using REST Client

Next, we will create a file for testing other requests like `POST`:

1. In Visual Studio Code, in the `RestClientTests` folder, create a file named `create-customer.http` and modify its contents to define a `POST` request to create a new customer, noting that REST Client will provide IntelliSense while you type common HTTP requests, as shown in the following code:

```
POST https://localhost:5001/api/customers/ HTTP/1.1
Content-Type: application/json
Content-Length: 301
{
  "customerID": "ABCXY",
  "companyName": "ABC Corp",
  "contactName": "John Smith",
  "contactTitle": "Sir",
  "address": "Main Street",
  "city": "New York",
  "region": "NY",
  "postalCode": "90210",
  "country":  "USA",
  "phone": "(123) 555-1234",
  "fax": null,
  "orders": null
}
```

1. Due to different line endings in different operating systems, the value for the `Content-Length` header will be different on Windows and macOS or Linux. If the value is wrong, then the request will fail. To discover the correct content length, select the body of the request and then look in the status bar for the number of characters, as shown in *Figure 15.5*:

Figure 15.5: Checking the correct Content-Length in the Visual Studio Code status bar

2. Send the request and note the response is `201 Created`. Also note the location (that is, the URL) of the newly created customer is https://localhost:5001/api/Customers/abcxy, and includes the newly created customer in the response body, as shown in *Figure 15.6*:

Figure 15.6: Adding a new customer by POSTing to the Web API service

I will leave you an optional challenge to create REST Client files that test updating a customer (using `PUT`) and deleting a customer (using `DELETE`). Try them on customers that do exist as well as customers that do not. Solutions are in the GitHub repository for this book.

Now that we've seen a quick and easy way to test our service, which also happens to be a great way to learn HTTP, what about external developers? We want it to be as easy as possible for them to learn and then call our service. For that purpose, we will use Swagger.

Understanding Swagger

The most important part of Swagger is the **OpenAPI Specification**, which defines a REST-style contract for your API, detailing all its resources and operations in a human- and machine-readable format for easy development, discovery, and integration.

Developers can use the OpenAPI Specification for a Web API to automatically generate strongly-typed client-side code in their preferred language or library.

For us, another useful feature is **Swagger UI**, because it automatically generates documentation for your API with built-in visual testing capabilities.

Let's review how Swagger is enabled for our web service using the `Swashbuckle` package:

1. If the web service is running, shut down the web server.
2. In `Northwind.WebApi.csproj`, note the package reference for `Swashbuckle.AspNetCore`, as shown in the following markup:

```
<ItemGroup>
  <PackageReference Include="Swashbuckle.AspNetCore" Version="6.2.3" />
</ItemGroup>
```

1. In `Program.cs`, import Swashbuckle's SwaggerUI namespace, as shown in the following code:

```
using Swashbuckle.AspNetCore.SwaggerUI; // SubmitMethod
```

1. In the section that configures the HTTP request pipeline, note the statements to use Swagger and Swagger UI when in development mode, and define an endpoint for the OpenAPI specification JSON document. Add code to explicitly list the HTTP methods that we want to support in our web service and change the endpoint name, as shown highlighted in the following code:

```
// Configure the HTTP request pipeline.
if (builder.Environment.IsDevelopment())
{
  app.UseSwagger();
  app.UseSwaggerUI(c =>
  {
    c.SwaggerEndpoint("/swagger/v1/swagger.json",
      "Northwind Service API Version 1");
    c.SupportedSubmitMethods(new[] {
      SubmitMethod.Get, SubmitMethod.Post,
      SubmitMethod.Put, SubmitMethod.Delete });
  });
}
```

Testing requests with Swagger UI

You are now ready to test an HTTP request using Swagger:

1. Start the `Northwind.WebApi` web service project.
2. In Chrome, navigate to `https://localhost:5001/swagger` and note that both the **Customers** and **WeatherForecast** Web API controllers have been discovered and documented, as well as **Schemas** used by the API.
3. Click **GET /api/Customers/{id}** to expand that endpoint and note the required parameter for the **id** of a customer.

4. Click **Try it out**, enter an **id** of `ALFKI`, and then click the wide blue **Execute** button, as shown in *Figure 15.7*:

Figure 15.7: Inputting a customer id before clicking the Execute button

5. Scroll down and note the **Request URL**, **Server response** with **Code**, and **Details** including **Response body** and **Response headers**, as shown in *Figure 15.8*:

Figure 15.8: Information on ALFKI in a successful Swagger request

6. Scroll back up to the top of the page, click **POST /api/Customers** to expand that section, and then click **Try it out**.
7. Click inside the **Request body** box, and modify the JSON to define a new customer, as shown in the following JSON:

```
{
  "customerID": "SUPER",
  "companyName": "Super Company",
  "contactName": "Rasmus Ibensen",
  "contactTitle": "Sales Leader",
  "address": "Rotterslef 23",
  "city": "Billund",
  "region": null,
  "postalCode": "4371",
  "country": "Denmark",
  "phone": "31 21 43 21",
  "fax": "31 21 43 22"
}
```

1. Click **Execute**, and note the **Request URL**, **Server response** with **Code**, and **Details** including **Response body** and **Response headers**, noting that a response code of `201` means the customer was successfully created, as shown in *Figure 15.9*:

Figure 15.9: Successfully adding a new customer

2. Scroll back up to the top of the page, click **GET /api/Customers**, click **Try it out**, enter `Denmark` for the country parameter, and click **Execute**, to confirm that the new customer was added to the database.

3. Click **DELETE /api/Customers/{id}**, click **Try it out**, enter `super` for the **id**, click **Execute**, and note that the **Server response Code** is `204`, indicating that it was successfully deleted, as shown in *Figure 15.10*:

Figure 15.10: Successfully deleting a customer

4. Click **Execute** again, and note that the **Server response Code** is `404`, indicating that the customer does not exist anymore, and the **Response body** contains a problem details JSON document, as shown in *Figure 15.11*:

Figure 15.11: The deleted customer does not exist anymore

5. Enter `bad` for the **id**, click **Execute** again, and note that the **Server response Code** is `400`, indicating that the customer did exist but failed to be deleted (in this case, because the web service is simulating this error), and the **Response body** contains a custom problem details JSON document, as shown in *Figure 15.12*:

Figure 15.12: The customer did exist but failed to be deleted

6. Use the `GET` methods to confirm that the new customer has been deleted from the database (there were originally only two customers in Denmark).
7. I will leave testing updates to an existing customer by using `PUT` to the reader.
8. Close Chrome and shut down the web server.

Consuming web services using HTTP clients

Now that we have built and tested our Northwind service, we will learn how to call it from any .NET app using the `HttpClient` class and its factory.

Understanding HttpClient

The easiest way to consume a web service is to use the `HttpClient` class. However, many people use it wrongly because it implements `IDisposable` and Microsoft's own documentation shows poor usage of it. See the book links in the GitHub repository for articles with more discussion of this.

Usually, when a type implements `IDisposable`, you should create it inside a `using` statement to ensure that it is disposed of as soon as possible. `HttpClient` is different because it is shared, reentrant, and partially thread-safe.

The problem has to do with how the underlying network sockets have to be managed. The bottom line is that you should use a single instance of it for each HTTP endpoint that you consume during the life of your application. This will allow each `HttpClient` instance to have defaults set that are appropriate for the endpoint it works with, while managing the underlying network sockets efficiently.

Configuring HTTP clients using HttpClientFactory

Microsoft is aware of the issue, and in ASP.NET Core 2.1 they introduced `HttpClientFactory` to encourage best practice; that is the technique we will use.

In the following example, we will use the Northwind MVC website as a client to the Northwind Web API service. Since both need to be hosted on a web server simultaneously, we first need to configure them to use different port numbers, as shown in the following list:

- The Northwind Web API service will listen on port `5002` using `HTTPS`.
- The Northwind MVC website will continue to listen on port `5000` using `HTTP` and port `5001` using `HTTPS`.

Let's configure those ports:

1. In the `Northwind.WebApi` project, in the `Properties` folder, in `launchSettings.json`, modify the `applicationUrl` setting to use post 5002, as shown in the following code:

```
"applicationUrl": "https://localhost:5002",
```

1. In the `Northwind.Mvc` project, in `Program.cs`, import the namespace for setting a media type header value, as shown in the following code:

```
using System.Net.Http.Headers; // MediaTypeWithQualityHeaderValue
```

1. In `Program.cs`, before calling the `Build` method, add a statement to enable `HttpClientFactory` with a named client to make calls to the Northwind Web API service using HTTPS on port `5002` and request JSON as the default response format, as shown in the following code:

```
builder.Services.AddHttpClient(name: "Northwind.WebApi",
  configureClient: options =>
  {
    options.BaseAddress = new Uri("https://localhost:5002/");
    options.DefaultRequestHeaders.Accept.Add(
      new MediaTypeWithQualityHeaderValue(
      mediaType: "application/json", quality: 1.0));
  });
```

Getting customers as JSON in the controller

We can now create an MVC controller action method that uses the factory to create an HTTP client, makes a `GET` request for customers, and deserializes the JSON response using convenience extension methods introduced with .NET 5 in the `System.Net.Http.Json` assembly and namespace:

1. In `Controllers/HomeController.cs`, declare a field to store the HTTP client factory, as shown in the following code:

```
private readonly IHttpClientFactory clientFactory;
```

1. Set the field in the constructor, as shown highlighted in the following code:

```
public HomeController(
  ILogger<HomeController> logger,
  NorthwindContext injectedContext,
  IHttpClientFactory httpClientFactory)
{
  _logger = logger;
  db = injectedContext;
  clientFactory = httpClientFactory;
}
```

1. Create a new action method for calling the Northwind Web API service, fetching all customers, and passing them to a view, as shown in the following code:

```
public async Task<IActionResult> Customers(string country)
{
  string uri;
  if (string.IsNullOrEmpty(country))
  {
```

```
    ViewData["Title"] = "All Customers Worldwide";
    uri = "api/customers";
  }
  else
  {
    ViewData["Title"] = $"Customers in {country}";
    uri = $"api/customers/?country={country}";
  }
  HttpClient client = clientFactory.CreateClient(
    name: "Northwind.WebApi");
  HttpRequestMessage request = new(
    method: HttpMethod.Get, requestUri: uri);
  HttpResponseMessage response = await client.SendAsync(request);
  IEnumerable<Customer>? model = await response.Content
    .ReadFromJsonAsync<IEnumerable<Customer>>();
  return View(model);
}
```

1. In the `Views/Home` folder, create a Razor file named `Customers.cshtml`.
2. Modify the Razor file to render the customers, as shown in the following markup:

```
@using Packt.Shared
@model IEnumerable<Customer>
<h2>@ViewData["Title"]</h2>
<table class="table">
  <thead>
    <tr>
      <th>Company Name</th>
      <th>Contact Name</th>
      <th>Address</th>
      <th>Phone</th>
    </tr>
  </thead>
  <tbody>
    @if (Model is not null)
    {
      @foreach (Customer c in Model)
      {
        <tr>
          <td>
            @Html.DisplayFor(modelItem => c.CompanyName)
          </td>
          <td>
            @Html.DisplayFor(modelItem => c.ContactName)
          </td>
          <td>
            @Html.DisplayFor(modelItem => c.Address)
            @Html.DisplayFor(modelItem => c.City)
            @Html.DisplayFor(modelItem => c.Region)
            @Html.DisplayFor(modelItem => c.Country)
            @Html.DisplayFor(modelItem => c.PostalCode)
          </td>
          <td>
            @Html.DisplayFor(modelItem => c.Phone)
          </td>
        </tr>
      }
    }
  </tbody>
</table>
```

1. In `Views/Home/Index.cshtml`, add a form after rendering the visitor count to allow visitors to enter a country and see the customers, as shown in the following markup:

```
<h3>Query customers from a service</h3>
<form asp-action="Customers" method="get">
  <input name="country" placeholder="Enter a country" />
  <input type="submit" />
</form>
```

1. Start the `Northwind.WebApi` project and confirm that the web service is listening only on port `5002`, as shown in the following output:

```
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: https://localhost:5002
```

1. Start the `Northwind.Mvc` project and confirm that the website is listening on ports `5000` and `5001`, as shown in the following output:

```
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: http://localhost:5000
```

1. Start Chrome.

2. On the home page, in the customer form, enter a country like `Germany`, `UK`, or `USA`, click **Submit**, and note the list of customers, as shown in *Figure 15.13*:

Figure 15.13: Customers in the UK

3. Click the **Back** button in your browser, clear the country textbox, click **Submit**, and note the worldwide list of customers.
4. In a command prompt or terminal, note the `HttpClient` writes each HTTP request that it makes and HTTP response that it receives, as shown in the following output:

```
info: System.Net.Http.HttpClient.Northwind.WebApi.ClientHandler[100]
  Sending HTTP request GET https://localhost:5002/api/customers/?country=UK
info: System.Net.Http.HttpClient.Northwind.WebApi.ClientHandler[101]
  Received HTTP response headers after 931.864ms - 200
```

1. Close Chrome and shut down the two web servers.

Implementing advanced features for web services

Now that you have seen the fundamentals of building a web service and then calling it from a client, let's look at some more advanced features.

Implementing a Health Check API

There are many paid services that perform site availability tests that are basic pings, some with more advanced analysis of the HTTP response.

ASP.NET Core 2.2 and later makes it easy to implement more detailed website health checks. For example, your website might be live, but is it ready? Can it retrieve data from its database?

Let's add basic health check capabilities to our web service:

1. In the `Northwind.WebApi` project, add a project reference to enable Entity Framework Core database health checks, as shown in the following markup:

```
<PackageReference Include=
  "Microsoft.Extensions.Diagnostics.HealthChecks.EntityFrameworkCore"
  Version="7.0.0" />
```

1. **Early Access Program**: Until the final release of .NET 7, the version numbers of packages will be for previews and release candidates. For example, `7.0.0-preview.1.22109.13`. You can change the version to `7.0.0-*` to automatically upgrade to the latest packages as monthly previews and release candidates are released.
2. Build the `Northwind.WebApi` project.
3. In `Program.cs`, before the call to `Build`, add a statement to add health checks, including to the Northwind database context, as shown in the following code:

```
builder.Services.AddHealthChecks()
  .AddDbContextCheck<NorthwindContext>();
```

1. By default, the database context check calls EF Core's `IDatabaseCreator.CanConnectAsync` method. You can customize what method is run by setting its name as a parameter in the `AddDbContextCheck` method.
2. In `Program.cs`, before the call to `MapControllers`, add a statement to use basic health checks, as shown in the following code:

```
app.UseHealthChecks(path: "/howdoyoufeel");
```

1. Start the `Northwind.WebApi` web service project.
2. Start Chrome.
3. Navigate to https://localhost:5002/howdoyoufeel and note that the web service responds with a plain text response: `Healthy`.
4. If you are using SQL Server, note the SQL statement that was executed to test the health of the database, as shown in the following output:

```
Level: Debug, Event Id: 20100, State: Executing DbCommand [Parameters=[], CommandType='Text', Cor
SELECT 1
```

1. Close Chrome and shut down the web server.

Implementing Open API analyzers and conventions

In this chapter, you learned how to enable Swagger to document a web service by manually decorating a controller class with attributes.

In ASP.NET Core 2.2 or later, there are API analyzers that reflect over controller classes that have been annotated with the `[ApiController]` attribute to document it automatically. The analyzer assumes some API conventions. After installing, controllers that have not been properly decorated should have warnings (green squiggles) and warnings when you compile the source code.

To use it, your project must enable the OpenAPI Analyzers:

1. In the `Northwind.WebApi` project file, add the element that enables the OpenAPI Analyzers, as shown highlighted in the following markup:

```
<PropertyGroup>
  <TargetFramework>net7.0</TargetFramework>
  <Nullable>enable</Nullable>
```

```
  <ImplicitUsings>enable</ImplicitUsings>
  <IncludeOpenAPIAnalyzers>true</IncludeOpenAPIAnalyzers>
</PropertyGroup>
```

1. In `CustomersController.cs`, for the `Update` action method, comment out the three `[ProducesResponseType]` attributes, as shown highlighted in the following code:

```
// [ProducesResponseType(204)]
// [ProducesResponseType(400)]
// [ProducesResponseType(404)]
public async Task<IActionResult> Update(
  string id, [FromBody] Customer c)
```

1. Note the warnings in **Error List** or **PROBLEMS**, as shown in *Figure 15.14*:

Figure 15.14: Warnings from the OpenAPI Analyzers

2. Uncomment the three attributes to remove the warnings.

Automatic code fixes can then add the appropriate `[Produces]` and `[ProducesResponseType]` attributes, although this only currently works in Visual Studio. In Visual Studio Code, you will see warnings about where the analyzer thinks you should add attributes, but you must add them yourself.

Implementing transient fault handling

When a client app or website calls a web service, it could be from across the other side of the world. Network problems between the client and the server could cause issues that are nothing to do with your implementation code. If a client makes a call and it fails, the app should not just give up. If it tries again, the issue may now have been resolved. We need a way to handle these temporary faults.

To handle these transient faults, Microsoft recommends that you use the third-party library Polly to implement automatic retries with exponential backoff. You define a policy, and the library handles everything else.

> **Good Practice**: You can read more about how Polly can make your web services more reliable at the following link: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/implement-resilient-applications/implement-http-call-retries-exponential-backoff-polly

Adding security HTTP headers

ASP.NET Core has built-in support for common security HTTP headers like HSTS. But there are many more HTTP headers that you should consider implementing.

The easiest way to add these headers is using a middleware class:

1. In the `Northwind.WebApi` project/folder, create a file named `SecurityHeadersMiddleware.cs` and modify its statements, as shown in the following code:

```
using Microsoft.Extensions.Primitives; // StringValues
public class SecurityHeaders
{
  private readonly RequestDelegate next;
  public SecurityHeaders(RequestDelegate next)
  {
    this.next = next;
  }
  public Task Invoke(HttpContext context)
  {
    // add any HTTP response headers you want here
    context.Response.Headers.Add(
      "super-secure", new StringValues("enable"));
    return next(context);
  }
}
```

1. In `Program.cs`, before the call to `MapControllers`, add a statement to register the middleware, as shown in the following code:

```
app.UseMiddleware<SecurityHeaders>();
```

1. Start the web service.
2. Start Chrome.
3. Show **Developer tools** and its **Network** tab to record requests and responses.
4. Navigate to https://localhost:5002/weatherforecast.
5. Note the custom HTTP response header that we added named `super-secure`, as shown in *Figure 15.15*:

Figure 15.15: Adding a custom HTTP header named super-secure

Enabling HTTP/3 support for HttpClient

in *Chapter 13*, *Building Websites Using ASP.NET Core Razor Pages*, you learned how to enable HTTP/3 support in the Kestrel web server. Now we will see how to enable HTTP/3 on the client-side. First, we need to enable all versions of HTTP in the web service project:

1. In the `Northwind.WebApi` project/folder, in `Program.cs`, import the namespace for working with HTTP protocols, as shown in the following code:

```
using Microsoft.AspNetCore.Server.Kestrel.Core; // HttpProtocols
```

1. In `Program.cs`, add statements before the call to `Build` to enable all three versions of HTTP, as shown in the following code:

```
builder.WebHost.ConfigureKestrel((context, options) =>
{
```

```
  options.ListenAnyIP(5002, listenOptions =>
  {
    listenOptions.Protocols = HttpProtocols.Http1AndHttp2AndHttp3;
    listenOptions.UseHttps(); // HTTP/3 requires secure connections
  });
});
```

1. In `appSettings.json`, add an entry to show hosting diagnostics, as shown highlighted in the following configuration:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning",
      "Microsoft.AspNetCore.Hosting.Diagnostics": "Information"
    }
```

1. In the `Northwind.Mvc` project/folder, add an element to enable HTTP/3 support, as shown in the following markup:

```
<ItemGroup>
  <RuntimeHostConfigurationOption
    Include="System.Net.SocketsHttpHandler.Http3Support"
    Value="true" />
</ItemGroup>
```

1. In `Program.cs`, add statements to the configuration of the HTTP client for the `Northwind.WebApi` service to set the default version to 3.0, and to fallback to earlier versions if HTTP/3 is not supported by the web service, as shown highlighted in the following code:

```
builder.Services.AddHttpClient(name: "Northwind.WebApi",
  configureClient: options =>
  {
    options.DefaultRequestVersion = HttpVersion.Version30;
    options.DefaultVersionPolicy = HttpVersionPolicy.RequestVersionOrLower;
    options.BaseAddress = new Uri("https://localhost:5002/");
    ...
  });
```

1. Start the `Northwind.WebApi` project and confirm that the web service is listening only on port `5002`, as shown in the following output:

```
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: https://localhost:5002
```

1. Start the `Northwind.Mvc` project.
2. Start Chrome and navigate to https://localhost:5001/.
3. On the home page, in the customer form, enter a country like `UK`, and then click **Submit**.
4. In the command line or terminal for the web service listening on port 5002, note the HTTP requests and responses are being made using HTTP/3, as shown in the following output:

```
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
      Request starting HTTP/3 GET https://localhost:5002/api/customers/?country=UK - -
```

1. Close Chrome and shut down the two web servers.

Building web services using minimal APIs

For .NET 6, Microsoft put a lot of effort into adding new features to the C# 10 language and simplifying the ASP.NET Core libraries to enable the creation of web services using minimal APIs.

You might remember the weather forecast service that is provided in the Web API project template. It shows the use of a controller class to return a five-day weather forecast using faked data. We will now recreate that weather service using minimal APIs.

It will listen on port `5003` and only `GET` requests are allowed:

1. Use your preferred code editor to open the `PracticalApps` solution/workspace and then add a new project, as defined in the following list:

   I. Project template: **ASP.NET Core Web API** / `webapi`
   II. Project file and folder: `Minimal.WebApi`
   III. Workspace/solution file and folder: `PracticalApps`
   IV. Other Visual Studio options: **Authentication Type**: None, **Configure for HTTPS**: selected, **Enable Docker**: cleared, **Use controllers (uncheck to use minimal APIs)**: cleared, **Enable OpenAPI support**: selected.

   Warning! If you are using Visual Studio 2022, make sure to clear the check box named **Use controllers** so that it uses minimal APIs instead. If you are using Visual Studio Code and the `dotnet new` command, use the `--use-minimal-apis` or `-minimal` switch.

2. In Visual Studio Code, select `Minimal.WebApi` as the active OmniSharp project.
3. Review `Program.cs`, as shown in the following code:

```
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
var app = builder.Build();
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
  app.UseSwagger();
  app.UseSwaggerUI();
}
app.UseHttpsRedirection();
var summaries = new[]
{
    "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scor
};
app.MapGet("/weatherforecast", () =>
{
  var forecast = Enumerable.Range(1, 5).Select(index =>
     new WeatherForecast
     (
         DateTime.Now.AddDays(index),
         Random.Shared.Next(-20, 55),
         summaries[Random.Shared.Next(summaries.Length)]
     ))
      .ToArray();
  return forecast;
})
.WithName("GetWeatherForecast");
app.Run();
internal record WeatherForecast(DateTime Date, int TemperatureC, string? Summary)
{
  public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);
}
```

Note the following:

- A record named `WeatherForecast` is defined at the bottom of `Program.cs` with three data store properties: `Date`, `TemperatureC`, `Summary`, and one calculated property: `TemperatureF`.
- The call to `MapGet` registers a handler for incoming GET requests to the relative path `/weatherforecast`. It returns fives `WeatherForecast` instances with random values for the `TemperatureC` and `Summary`

properties.
- There is no `Controller` folder or controller classes.
- **Good Practice**: For simple web services, avoid creating a controller class, and instead use minimal APIs to put all the configuration and implementation in one place, `Program.cs`.

1. In the `Properties` folder, in `launchSettings.json`, configure the `Minimal.WebApi` profile to launch the browser using port `5003` in the URL and the relative API path, as shown highlighted in the following markup:

```
"profiles": {
  "Minimal.WebApi": {
    "commandName": "Project",
    "dotnetRunMessages": "true",
    "launchBrowser": true,
    "launchUrl": "weatherforecast",
    "applicationUrl": "https://localhost:5003",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
```

Testing the minimal weather service

Before creating a client to the service, let's test that it returns forecasts as JSON:

1. Start the `Minimal.WebApi` web service project.
2. If you are not using Visual Studio 2022, start Chrome and navigate to https://localhost:5003/weatherforecast.
3. Note the Web API service should return a JSON document with five random weather forecast objects in an array.
4. Close Chrome and shut down the web server.

Adding weather forecasts to the Northwind website home page

Finally, let's add an HTTP client to the Northwind website so that it can call the weather service and show forecasts on the home page:

1. In the `Northwind.Mvc` project, in the `Models` folder, add a class file named `WeatherForecast.cs`, as shown in the following code:

```
namespace Northwind.Mvc.Models;
public record WeatherForecast(DateTime Date, int TemperatureC, string? Summary)
{
  public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);
}
```

1. In `Program.cs`, before the call to `Build`, add a statement to configure an HTTP client to call the minimal service on port `5003`, as shown in the following code:

```
builder.Services.AddHttpClient(name: "Minimal.WebApi",
  configureClient: options =>
  {
    options.BaseAddress = new Uri("https://localhost:5003/");
    options.DefaultRequestHeaders.Accept.Add(
      new MediaTypeWithQualityHeaderValue(
      "application/json", 1.0));
  });
```

1. In `HomeController.cs`, in the `Index` action method, before the call to `View`, add statements to get and use an HTTP client to call the weather service to get forecasts and store them in `ViewData`, as shown in the following code:

```
try
{
```

```
  HttpClient client = clientFactory.CreateClient(
    name: "Minimal.WebApi");
  HttpRequestMessage request = new(
    method: HttpMethod.Get, requestUri: "weatherforecast");
  HttpResponseMessage response = await client.SendAsync(request);
  ViewData["weather"] = await response.Content
    .ReadFromJsonAsync<WeatherForecast[]>();
}
catch (Exception ex)
{
  _logger.LogWarning(
    $"The Minimal.WebApi service is not responding. Exception: {ex.Message}");
  ViewData["weather"] = Enumerable.Empty<WeatherForecast>().ToArray();
}
```

1. In `Views/Home`, in `Index.cshtml`, in the top code block get the weather forecasts from the `ViewData` dictionary, as shown highlighted in the following markup:

```
@{
  ViewData["Title"] = "Home Page";
  string currentItem = "";
  WeatherForecast[]? weather = ViewData["weather"] as WeatherForecast[];
}
```

1. In the first `<div>`, after rendering the current time, add markup to enumerate the weather forecasts unless there aren't any, and render them in a table, as shown in the following markup:

```
<p>
  <h4>Five-Day Weather Forecast</h4>
  @if ((weather is null) || (!weather.Any()))
  {
    <p>No weather forecasts found.</p>
  }
  else
  {
  <table class="table table-info">
    <tr>
      @foreach (WeatherForecast w in weather)
      {
        <td>@w.Date.ToString("ddd d MMM") will be @w.Summary</td>
      }
    </tr>
  </table>
  }
</p>
```

1. Start the `Minimal.WebApi` web service project.
2. Start the `Northwind.Mvc` website project.

3. Navigate to https://localhost:5001/, and note the weather forecast, as shown in *Figure 15.16*:

Figure 15.16: A five-day weather forecast on the home page of the Northwind website

4. View the command prompt or terminal for the MVC website and note the info messages that indicate a request was sent to the minimal API web service `api/weather` endpoint in about 83ms, as shown in the following output:

```
info: System.Net.Http.HttpClient.Minimal.WebApi.LogicalHandler[100]
      Start processing HTTP request GET https://localhost:5003/weatherforecast
info: System.Net.Http.HttpClient.Minimal.WebApi.ClientHandler[100]
      Sending HTTP request GET https://localhost:5003/weatherforecast
info: System.Net.Http.HttpClient.Minimal.WebApi.ClientHandler[101]
      Received HTTP response headers after 76.8963ms - 200
info: System.Net.Http.HttpClient.Minimal.WebApi.LogicalHandler[101]
      End processing HTTP request after 82.9515ms - 200
```

1. Stop the `Minimal.WebApi` service, refresh the browser, and note that after a few seconds the MVC website home page appears without weather forecasts because the web service is not responding.
2. Close Chrome and shut down the web server.

Practicing and exploring

Test your knowledge and understanding by answering some questions, get some hands-on practice, and explore this chapter's topics with deeper research.

Exercise 15.1 – Test your knowledge

Answer the following questions:

1. Which class should you inherit from to create a controller class for an ASP.NET Core Web API service?
2. If you decorate your controller class with the `[ApiController]` attribute to get default behavior like automatic `400` responses for invalid models, what else must you do?
3. What must you do to specify which controller action method will be executed in response to an HTTP request?

4. What must you do to specify what responses should be expected when calling an action method?
5. List three methods that can be called to return responses with different status codes.
6. List four ways that you can test a web service.
7. Why should you not wrap your use of `HttpClient` in a `using` statement to dispose of it when you are finished even though it implements the `IDisposable` interface, and what should you use instead?
8. What does the acronym CORS stand for and why is it important to enable it in a web service?
9. How can you enable clients to detect if your web service is healthy with ASP.NET Core 2.2 and later?
10. What benefits does endpoint routing provide?

Exercise 15.2 – Practice creating and deleting customers with HttpClient

Extend the `Northwind.Mvc` website project to have pages where a visitor can fill in a form to create a new customer, or search for a customer and then delete them. The MVC controller should make calls to the Northwind web service to create and delete customers.

Exercise 15.3 – Explore topics

Use the links on the following page to learn more detail about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-15---building-and-consuming-web-services

Summary

In this chapter, you learned how to build an ASP.NET Core Web API service that can be called by any app on any platform that can make an HTTP request and process an HTTP response.

You also learned how to test and document web service APIs with Swagger, as well as how to consume services efficiently.

In the next chapter, you will learn to build user interfaces using Blazor, Microsoft's component technology that enables developers to build client-side, single-page applications (SPAs) for websites using C# instead of JavaScript, hybrid apps for desktop, and potentially mobile apps.

16 Building User Interfaces Using Blazor

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter is about using Blazor to build user interfaces. I will describe the different flavors of Blazor and their pros and cons.

You will learn how to build Blazor components that can execute their code on the web server or in the web browser. When hosted with Blazor Server, it uses SignalR to communicate needed updates to the user interface in the browser. When hosted with Blazor WebAssembly, the components execute their code in the client and must make HTTP calls to interact with the server.

In this chapter, we will cover the following topics:

- Understanding Blazor
- Comparing Blazor project templates
- Building components using Blazor Server
- Building components using Blazor WebAssembly
- Improving Blazor WebAssembly apps

Understanding Blazor

Blazor lets you build shared components and interactive web user interfaces using C# instead of JavaScript. In April 2019, Microsoft announced that Blazor "is no longer experimental and we are committing to ship it as a supported web UI framework, including support for running client side in the browser on WebAssembly." Blazor is supported on all modern browsers.

JavaScript and friends

Traditionally, any code that needs to execute in a web browser is written using the JavaScript programming language or a higher-level technology that **transpiles** (transforms or compiles) into JavaScript. This is because all browsers have supported JavaScript for about two decades, so it has become the lowest common denominator for implementing business logic on the client side.

JavaScript does have some issues, however. Although it has superficial similarities to C-style languages like C# and Java, it is actually very different once you dig beneath the surface. It is a dynamically typed pseudo-functional language that uses prototypes instead of class inheritance for object reuse. It might look human, but you will get a surprise when it's revealed to be a Skrull.

Wouldn't it be great if we could use the same language and libraries in a web browser as we do on the server side?

Silverlight – C# and .NET using a plugin

Microsoft made a previous attempt at achieving this goal with a technology named Silverlight. When Silverlight 2.0 was released in 2008, a C# and .NET developer could use their skills to build libraries and visual components

that were executed in the web browser by the Silverlight plugin.

By 2011 and Silverlight 5.0, Apple's success with the iPhone and Steve Jobs' hatred of browser plugins like Flash eventually led to Microsoft abandoning Silverlight since, like Flash, Silverlight is banned from iPhones and iPads.

WebAssembly – a target for Blazor

A recent development in browsers has given Microsoft the opportunity to make another attempt. In 2017, the **WebAssembly Consensus** was completed, and all major browsers now support it: Chromium (Chrome, Edge, Opera, Brave), Firefox, and WebKit (Safari). Although Blazor Server is supported on Internet Explorer 11, Blazor WebAssembly is not.

WebAssembly (**Wasm**) is a binary instruction format for a virtual machine that provides a way to run code written in multiple languages on the web at near-native speed. Wasm is designed as a portable target for the compilation of high-level languages like C#.

Understanding Blazor hosting models

Blazor is a single programming or app model with multiple hosting models:

- **Blazor Server** runs on the server side, like Razor Pages and ASP.NET Core MVC, so the C# code that you write has full access to all resources that your business logic might need without needing to supply credentials to authenticate. It then uses SignalR to communicate user interface updates to the client side. The server must keep a live SignalR connection to each client and track the current state of every client, so Blazor Server does not scale well if you need to support lots of clients. It first shipped as part of ASP.NET Core 3.0 in September 2019.
- **Blazor WebAssembly** runs on the client side, so the C# code that you write only has access to resources in the browser and it must make HTTP calls (that might require authentication) before it can access resources on the server. It first shipped as an extension to ASP.NET Core 3.1 in May 2020 and was versioned 3.2 because it was a Current release and therefore not covered by ASP.NET Core 3.1's Long Term Support. The Blazor WebAssembly 3.2 version used the Mono runtime and Mono libraries; .NET 5 and later versions use the Mono runtime and the .NET libraries.
- **.NET MAUI Blazor App**, aka **Blazor Hybrid**, runs in the .NET process, renders its web UI to a web view control using a local interop channel, and is hosted in a .NET MAUI app. It is conceptually like Electron apps that use Node.js.

This multi-host model means that, with careful planning, a developer can write Blazor components once, and then run them on the web server side, web client side, or within a desktop app.

Understanding Blazor components

It is important to understand that Blazor is used to create **user interface components**. Components define how to render the user interface, react to user events, and can be composed and nested, and compiled into a Razor class library for packaging and distribution.

For example, to provide a user interface for star ratings of products on a commerce site, you might create a component named `Rating.razor`, as shown in the following markup:

```
<div>
@for (int i = 0; i < Maximum; i++)
{
  if (i < Value)
  {
    <span class="oi oi-star-filled" />
  }
  else
  {
    <span class="oi oi-star-empty" />
  }
```

```
}
</div>
@code {
  [Parameter]
  public byte Maximum { get; set; }
  [Parameter]
  public byte Value { get; set; }
}
```

You could then use the component on a web page, as shown in the following markup:

```
<h1>Review</h1>
<Rating id="rating" Maximum="5" Value="3" />
<textarea id="comment" />
```

Instead of a single file with both markup and an `@code` block, the code can be stored in a separate code-behind file named `Rating.razor.cs`. The class in this file must be `partial` and have the same name as the component.

There are many built-in Blazor components, including ones to set elements like `<title>` in the `<head>` section of a web page, and plenty of third parties who will sell you components for common purposes.

In the future, Blazor might not be limited to only creating user interface components using web technologies. Microsoft has an experimental technology known as **Blazor Mobile Bindings** that allows developers to use Blazor to build mobile user interface components. Instead of using HTML and CSS to build a web user interface, it uses XAML and .NET MAUI to build a cross-platform graphical user interface.

What is the difference between Blazor and Razor?

You might wonder why Blazor components use `.razor` as their file extension. Razor is a template markup syntax that allows the mixing of HTML and C#. Older technologies that support Razor syntax use the `.cshtml` file extension to indicate the mix of C# and HTML.

Razor syntax is used for:

- ASP.NET Core MVC **views** and **partial views** that use the `.cshtml` file extension. The business logic is separated into a controller class that treats the view as a template to push the view model to, which then outputs it to a web page.
- **Razor Pages** that use the `.cshtml` file extension. The business logic can be embedded or separated into a file that uses the `.cshtml.cs` file extension. The output is a web page.
- **Blazor components** that use the `.razor` file extension. The output is not a web page, although layouts can be used to wrap a component so it outputs as a web page, and the `@page` directive can be used to assign a route that defines the URL path to retrieve the component as a page.

Comparing Blazor project templates

One way to understand the choice between the Blazor Server and Blazor WebAssembly hosting models is to review the differences in their default project templates.

Reviewing the Blazor Server project template

Let us look at the default template for a Blazor Server project. Mostly you will see that it is the same as an ASP.NET Core Razor Pages template, with a few key additions:

1. Use your preferred code editor to open the `PracticalApps` solution/workspace and then add a new project, as defined in the following list:
   I. Project template: **Blazor Server App** / `blazorserver`
   II. Workspace/solution file and folder: `PracticalApps`
   III. Project file and folder: `Northwind.BlazorServer`

IV. Other Visual Studio options: **Authentication Type**: **None**; **Configure for HTTPS**: selected; **Enable Docker**: cleared

2. In Visual Studio Code, select `Northwind.BlazorServer` as the active OmniSharp project.
3. Build the `Northwind.BlazorServer` project.
4. In `Northwind.BlazorServer.csproj`, note that it is identical to an ASP.NET Core project that uses the Web SDK and targets .NET 7.0.
5. In `Program.cs`, note it is almost identical to an ASP.NET Core project. Differences include the section that configures services, with its call to the `AddServerSideBlazor` method, as shown highlighted in the following code:

```
builder.Services.AddRazorPages();
builder.Services.AddServerSideBlazor();
builder.Services.AddSingleton<WeatherForecastService>();
```

1. Also note the section for configuring the HTTP pipeline, which adds the calls to the `MapBlazorHub` and `MapFallbackToPage` methods that configure the ASP.NET Core app to accept incoming SignalR connections for Blazor components, while other requests fall back to a Razor Page named `_Host.cshtml`, as shown highlighted in the following code:

```
app.UseRouting();
app.MapBlazorHub();
app.MapFallbackToPage("/_Host");
app.Run();
```

1. In the `Pages` folder, open `_Host.cshtml` and note that it sets a shared layout named `_Layout` and renders a Blazor component of type `App` that is prerendered on the server, as shown in the following markup:

```
@page "/"
@namespace  Northwind.BlazorServer.Pages
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@{
  Layout = "_Layout";
}
<component type="typeof(App)" render-mode="ServerPrerendered" />
```

1. In the `Pages` folder, open the shared layout file named `_Layout.cshtml`, as shown in the following markup:

```
@using Microsoft.AspNetCore.Components.Web
@namespace Northwind.BlazorServer.Pages
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport"
        content="width=device-width, initial-scale=1.0" />
  <base href="~/" />
  <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
  <link href="css/site.css" rel="stylesheet" />
  <link href="Northwind.BlazorServer.styles.css" rel="stylesheet" />
  <component type="typeof(HeadOutlet)" render-mode="ServerPrerendered" />
</head>
<body>
  @RenderBody()
  <div id="blazor-error-ui">
    <environment include="Staging,Production">
      An error has occurred. This application may no longer respond until reloaded.
    </environment>
    <environment include="Development">
      An unhandled exception has occurred. See browser dev tools for details.
    </environment>
    <a href="" class="reload">Reload</a>
    <a class="dismiss">✕</a>
  </div>
  <script src="_framework/blazor.server.js"></script>
```

```
</body>
</html>
```

While reviewing the preceding markup, note the following:

- `<div id="blazor-error-ui">` for showing Blazor errors that will appear as a yellow bar at the bottom of the web page when an error occurs
- The script block for `blazor.server.js` manages the SignalR connection back to the server

1. In the `Northwind.BlazorServer` folder, open `App.razor` and note that it defines a `Router` for all components found in the current assembly, as shown in the following code:

```
<Router AppAssembly="@typeof(App).Assembly">
  <Found Context="routeData">
    <RouteView RouteData="@routeData"
               DefaultLayout="@typeof(MainLayout)" />
    <FocusOnNavigate RouteData="@routeData" Selector="h1" />
  </Found>
  <NotFound>
    <PageTitle>Not found</PageTitle>
    <LayoutView Layout="@typeof(MainLayout)">
      <p>Sorry, there's nothing at this address.</p>
    </LayoutView>
  </NotFound>
</Router>
```

While reviewing the preceding markup, note the following:

- If a matching route is found, then `RouteView` is executed that sets the default layout for the component to `MainLayout` and passes any route data parameters to the component.
- If a matching route is not found, then `LayoutView` is executed that renders the internal markup (in this case, a simple paragraph element with a message telling the visitor there is nothing at this address) inside `MainLayout`.

1. In the `Shared` folder, open `MainLayout.razor` and note that it defines `<div>` for a sidebar containing a navigation menu that is implemented by the `NavMenu.razor` component file in this project, and an HTML5 element such as `<main>` and `<article>` for the content, as shown in the following code:

```
@inherits LayoutComponentBase
<PageTitle>Northwind.BlazorServer</PageTitle>
<div class="page">
  <div class="sidebar">
    <NavMenu />
  </div>
  <main>
    <div class="top-row px-4">
      <a href="https://docs.microsoft.com/aspnet/"
         target="_blank">About</a>
    </div>
    <article class="content px-4">
      @Body
    </article>
  </main>
</div>
```

1. In the `Shared` folder, open `MainLayout.razor.css` and note that it contains isolated CSS styles for the component. Due to the naming convention, styles defined in this file take priority over others defined elsewhere that might affect the component.
2. In the `Shared` folder, open `NavMenu.razor` and note that it has three menu items for **Home**, **Counter**, and **Fetch data**. These are created by using a Microsoft-provided Blazor component named `NavLink`, as shown in the following markup:

```
<div class="top-row ps-3 navbar navbar-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="">Northwind.BlazorServer</a>
```

```
    <button title="Navigation menu" class="navbar-toggler"
            @onclick="ToggleNavMenu">
      <span class="navbar-toggler-icon"></span>
    </button>
  </div>
</div>
<div class="@NavMenuCssClass" @onclick="ToggleNavMenu">
  <nav class="flex-column">
    <div class="nav-item px-3">
      <NavLink class="nav-link" href="" Match="NavLinkMatch.All">
        <span class="oi oi-home" aria-hidden="true"></span> Home
      </NavLink>
    </div>
    <div class="nav-item px-3">
      <NavLink class="nav-link" href="counter">
        <span class="oi oi-plus" aria-hidden="true"></span> Counter
      </NavLink>
    </div>
    <div class="nav-item px-3">
      <NavLink class="nav-link" href="fetchdata">
        <span class="oi oi-list-rich" aria-hidden="true"></span> Fetch data
      </NavLink>
    </div>
  </nav>
</div>
@code {
  private bool collapseNavMenu = true;
  private string? NavMenuCssClass => collapseNavMenu ? "collapse" : null;
  private void ToggleNavMenu()
  {
    collapseNavMenu = !collapseNavMenu;
  }
}
```

1. In the `Pages` folder, open `FetchData.razor` and note that it defines a component that fetches weather forecasts from an injected dependency weather service and then renders them in a table, as shown in the following code:

```
@page "/fetchdata"
<PageTitle>Weather forecast</PageTitle>
@using Northwind.BlazorServer.Data
@inject WeatherForecastService ForecastService
<h1>Weather forecast</h1>
<p>This component demonstrates fetching data from a service.</p>
@if (forecasts == null)
{
  <p><em>Loading...</em></p>
}
else
{
  <table class="table">
    <thead>
      <tr>
        <th>Date</th>
        <th>Temp. (C)</th>
        <th>Temp. (F)</th>
        <th>Summary</th>
      </tr>
    </thead>
    <tbody>
    @foreach (var forecast in forecasts)
    {
      <tr>
        <td>@forecast.Date.ToShortDateString()</td>
        <td>@forecast.TemperatureC</td>
        <td>@forecast.TemperatureF</td>
        <td>@forecast.Summary</td>
       </tr>
    }
    </tbody>
```

```
  </table>
}
@code {
  private WeatherForecast[]? forecasts;
  protected override async Task OnInitializedAsync()
  {
    forecasts = await ForecastService.GetForecastAsync(DateTime.Now);
  }
}
```

1. In the `Data` folder, open `WeatherForecastService.cs` and note that it is *not* a Web API controller class; it is just an ordinary class that returns random weather data, as shown in the following code:

```
namespace Northwind.BlazorServer.Data
{
  public class WeatherForecastService
  {
    private static readonly string[] Summaries = new[]
    {
      "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm",
      "Balmy", "Hot", "Sweltering", "Scorching"
    };
    public Task<WeatherForecast[]> GetForecastAsync(DateTime startDate)
    {
      return Task.FromResult(Enumerable.Range(1, 5)
        .Select(index => new WeatherForecast
          {
            Date = startDate.AddDays(index),
            TemperatureC = Random.Shared.Next(-20, 55),
            Summary = Summaries[Random.Shared.Next(Summaries.Length)]
          }).ToArray());
    }
  }
}
```

Understanding CSS and JavaScript isolation

Blazor components often need to provide their own CSS to apply styling or JavaScript for activities that cannot be performed purely in C#, like access to browser APIs. To ensure this does not conflict with site-level CSS and JavaScript, Blazor supports CSS and JavaScript isolation. If you have a component named `Index.razor`, simply create a CSS file named `Index.razor.css`. The styles defined within this file will override any other styles in the project.

Understanding Blazor routing to page components

The `Router` component that we saw in the `App.razor` file enables routing to components. The markup for creating an instance of a component looks like an HTML tag where the name of the tag is the component type. Components can be embedded on a web page using an element, for example, `<Rating Stars="5" />`, or can be routed to like a Razor Page or MVC controller.

How to define a routable page component

To create a routable page component, add the `@page` directive to the top of a component's `.razor` file, as shown in the following markup:

```
@page "customers"
```

The preceding code is the equivalent of an MVC controller decorated with the `[Route]` attribute, as shown in the following code:

```
[Route("customers")]
public class CustomersController
{
```

The `Router` component scans the assembly specifically in its `AppAssembly` parameter for components decorated with the `[Route]` attribute and registers their URL paths.

Any single-page component can have multiple `@page` directives to register multiple routes.

At runtime, the page component is merged with any specific layout that you have specified, just like an MVC view or Razor Page would be. By default, the Blazor Server project template defines `MainLayout.razor` as the layout for page components.

> **Good Practice**: By convention, put routable page Blazor components in the `Pages` folder.

How to navigate Blazor routes

Microsoft provides a dependency service named `NavigationManager` that understands Blazor routing and the `NavLink` component. The `NavigateTo` method is used to go to the specified URL.

How to pass route parameters

Blazor routes can include case-insensitive named parameters, and your code can most easily access the passed values by binding the parameter to a property in the code block using the `[Parameter]` attribute, as shown in the following markup:

```
@page "/customers/{country}"
<div>Country parameter as the value: @Country</div>
@code {
  [Parameter]
  public string Country { get; set; }
}
```

The recommended way to handle a parameter that should have a default value when it is missing is to suffix the parameter with `?` and use the null coalescing operator in the `OnParametersSet` method, as shown in the following markup:

```
@page "/customers/{country?}"
<div>Country parameter as the value: @Country</div>
@code {
  [Parameter]
  public string Country { get; set; }
  protected override void OnParametersSet()
  {
    // if the automatically set property is null
    // set its value to USA
    Country = Country ?? "USA";
  }
}
```

Understanding base component classes

The `OnParametersSet` method is defined by the base class that components inherit from by default named `ComponentBase`, as shown in the following code:

```
using Microsoft.AspNetCore.Components;
public abstract class ComponentBase : IComponent, IHandleAfterRender, IHandleEvent
{
  // members not shown
}
```

`ComponentBase` has some useful methods that you can call and override, as shown in the following table:

Method(s)	Description
`InvokeAsync`	Call this method to execute a function on the associated

	renderer's synchronization context.
`OnAfterRender`, `OnAfterRenderAsync`	Override these methods to invoke code after each time the component has been rendered.
`OnInitialized`, `OnInitializedAsync`	Override these methods to invoke code after the component has received its initial parameters from its parent in the render tree.
`OnParametersSet`, `OnParametersSetAsync`	Override these methods to invoke code after the component has received parameters and the values have been assigned to properties.
`ShouldRender`	Override this method to indicate if the component should render.
`StateHasChanged`	Call this method to cause the component to re-render.

Blazor components can have shared layouts in a similar way to MVC views and Razor Pages. You would create a `.razor` component file and make it explicitly inherit from `LayoutComponentBase`, as shown in the following markup:

```
@inherits LayoutComponentBase
<div>
  ...
  @Body
  ...
</div>
```

The base class has a property named `Body` that you can render in the markup at the correct place within the layout.

You can set a default layout for components in the `App.razor` file and its `Router` component. To explicitly set a layout for a component, use the `@layout` directive, as shown in the following markup:

```
@page "/customers"
@layout AlternativeLayout
<div>
  ...
</div>
```

How to use the navigation link component with routes

In HTML, you use the `<a>` element to define navigation links, as shown in the following markup:

```
<a href="/customers">Customers</a>
```

In Blazor, use the `<NavLink>` component, as shown in the following markup:

```
<NavLink href="/customers">Customers</NavLink>
```

The `NavLink` component is better than an anchor element because it automatically sets its class to `active` if its `href` is a match on the current location URL. If your CSS uses a different class name, then you can set the class name in the `NavLink.ActiveClass` property.

By default, in the matching algorithm, the `href` is a path *prefix*, so if `NavLink` has an `href` of `/customers`, as shown in the preceding code example, then it would match all the following paths and set them all to have the `active` class style:

```
/customers
/customers/USA
/customers/Germany/Berlin
```

To ensure that the matching algorithm only performs matches on *all* the paths, set the `Match` parameter to `NavLinkMatch.All`, as shown in the following code:

```
<NavLink href="/customers" Match="NavLinkMatch.All">Customers</NavLink>
```

If you set other attributes such as `target`, they are passed through to the underlying `<a>` element that is generated.

Running the Blazor Server project template

Now that we have reviewed the project template and the important parts that are specific to Blazor Server, we can start the website and review its behavior:

1. In the `Properties` folder, in `launchSettings.json`, modify the `applicationUrl` to use port `5004` for `HTTP` and port `5005` for `HTTPS`, as shown highlighted in the following markup:

```
"profiles": {
  "Northwind.BlazorServer": {
    "commandName": "Project",
    "dotnetRunMessages": true,
    "launchBrowser": true,
    "applicationUrl": "https://localhost:5005;http://localhost:5004",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
```

1. Start the website.
2. Start Chrome and navigate to `https://localhost:5005/`.

3. In the left navigation menu, click **Fetch data**, as shown in *Figure 16.1*:

Figure 16.1: Fetching weather data into a Blazor Server app

4. In the browser address bar, change the route to `/apples` and note the missing message, as shown in *Figure 16.2*:

Figure 16.2: The missing component message

5. Close Chrome and shut down the web server.

Reviewing the Blazor WebAssembly project template

Now we will create a Blazor WebAssembly project. I will not show its code in the book if the code is the same as in a Blazor Server project:

1. Use your preferred code editor to open the `PracticalApps` solution/workspace and then add a new project, as defined in the following list:
   I. Project template: **Blazor WebAssembly App** / `blazorwasm`
   II. Project file and folder: `Northwind.BlazorWasm`
   III. `dotnet new` switches: `--pwa --hosted`
   IV. Workspace/solution file and folder: `PracticalApps`
   V. **Authentication Type**: **None**
   VI. **Configure for HTTPS**: checked
   VII. **ASP.NET Core hosted**: checked
   VIII. **Progressive Web Application**: checked

While reviewing the generated folders and files, note that *three* projects are generated, as described in the following list:

- `Northwind.BlazorWasm.Client` is a Blazor WebAssembly project stored in the `Northwind.BlazorWasm\Client` folder.
- `Northwind.BlazorWasm.Server` is an ASP.NET Core project website stored in the `Northwind.BlazorWasm\Server` folder for hosting the weather service that has the same implementation for returning random weather forecasts as before, but is implemented as a proper Web API controller class. The project file has project references to `Shared` and `Client`, and a package reference to support Blazor WebAssembly on the server side.
- `Northwind.BlazorWasm.Shared` is a class library stored in the `Northwind.BlazorWasm\Shared` folder that contains models for the weather service.

The folder structure is simplified, as shown in *Figure 16.3*:

Figure 16.3: The folder structure for the Blazor WebAssembly project template

Understanding deployment choices for Blazor WebAssembly apps

There are two main ways to deploy a Blazor WebAssembly app:

- You could deploy just the `Client` project by placing its published files in any static hosting web server. It could be configured to call the weather service that you created in *Chapter 15*, *Building and Consuming Web Services*.
- You could deploy the `Server` project, which references the `Client` app and hosts both the weather service and the Blazor WebAssembly app. The Blazor WebAssembly app is placed in the server website `wwwroot` folder along with any other static assets.

More Information: You can read more about these choices at the following link: https://docs.microsoft.com/en-us/aspnet/core/blazor/host-and-deploy/webassembly

Differences between Blazor Server and Blazor WebAssembly projects

Now let us review the differences between a Blazor Server and a Blazor WebAssembly project:

1. In the `Client` folder, in `Northwind.BlazorWasm.Client.csproj`, note that it uses the Blazor WebAssembly SDK and references two WebAssembly packages and the `Shared` project, as well as the service worker required for PWA support, as shown in the following markup:

```
<Project Sdk="Microsoft.NET.Sdk.BlazorWebAssembly">
  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <ServiceWorkerAssetsManifest>service-worker-assets.js
      </ServiceWorkerAssetsManifest>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include=
      "Microsoft.AspNetCore.Components.WebAssembly"
      Version="7.0.0" />
    <PackageReference Include=
      "Microsoft.AspNetCore.Components.WebAssembly.DevServer"
      Version="7.0.0" PrivateAssets="all" />
  </ItemGroup>
  <ItemGroup>
    <ProjectReference Include=
      "..\Shared\Northwind.BlazorWasm.Shared.csproj" />
```

```
  </ItemGroup>
  <ItemGroup>
    <ServiceWorker Include="wwwroot\service-worker.js"
      PublishedContent="wwwroot\service-worker.published.js" />
  </ItemGroup>
</Project>
```

Early Access Program: Until the final release of .NET 7, the version numbers of packages will be for previews and release candidates. For example, `7.0.0-preview.1.22109.13`. You can change the version to `7.0.0-*` to automatically upgrade to the latest packages as monthly previews and release candidates are released.

1. In the `Client` folder, in `Program.cs`, note that the host builder is for `WebAssembly` instead of server-side ASP.NET Core, and that it registers a dependency service for making HTTP requests, which is an extremely common requirement for Blazor WebAssembly apps, as shown in the following code:

```
using Microsoft.AspNetCore.Components.Web;
using Microsoft.AspNetCore.Components.WebAssembly.Hosting;
using Northwind.BlazorWasm.Client;
var builder = WebAssemblyHostBuilder.CreateDefault(args);
builder.RootComponents.Add<App>("#app");
builder.RootComponents.Add<HeadOutlet>("head::after");
builder.Services.AddScoped(sp => new HttpClient
  { BaseAddress = new Uri(builder.HostEnvironment.BaseAddress) });
await builder.Build().RunAsync();
```

1. In the `wwwroot` folder, in `index.xhtml`, note the `manifest.json` and `service-worker.js` files supporting offline work, and the `blazor.webassembly.js` script that downloads all the NuGet packages for Blazor WebAssembly, as shown in the following markup:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-s
  <title>Northwind.BlazorWasm</title>
  <base href="/" />
  <link href="css/bootstrap/bootstrap.min.css" rel="stylesheet" />
  <link href="css/app.css" rel="stylesheet" />
  <link href="Northwind.BlazorWasm.Client.styles.css" rel="stylesheet" />
  <link href="manifest.json" rel="manifest" />
  <link rel="apple-touch-icon" sizes="512x512" href="icon-512.png" />
  <link rel="apple-touch-icon" sizes="192x192" href="icon-192.png" />
</head>
<body>
  <div id="app">Loading...</div>
  <div id="blazor-error-ui">
    An unhandled error has occurred.
    <a href="" class="reload">Reload</a>
    <a class="dismiss">✕</a>
  </div>
  <script src="_framework/blazor.webassembly.js"></script>
  <script>navigator.serviceWorker.register('service-worker.js');</script>
</body>
</html>
```

1. In the `Pages` folder, open `FetchData.razor` and note that the markup is like Blazor Server except for the injected dependency service for making HTTP requests, as shown highlighted in the following partial markup:

```
@page "/fetchdata"
@using Northwind.BlazorWasm.Shared
@inject HttpClient Http
<PageTitle>Weather forecast</PageTitle>
<h1>Weather forecast</h1>
...
@code {
```

```
  private WeatherForecast[]? forecasts;
  protected override async Task OnInitializedAsync()
  {
    forecasts = await
      Http.GetFromJsonAsync<WeatherForecast[]>("WeatherForecast");
  }
}
```

1. In the `Northwind.BlazorWasm.Server` project, in the `Properties` folder, in `launchSettings.json`, modify the `applicationUrl` to use port `5006` for `HTTP` and port `5007` for `HTTPS`, as shown highlighted in the following markup:

```
"profiles": {
  "Northwind.BlazorServer": {
    "commandName": "Project",
    "dotnetRunMessages": true,
    "launchBrowser": true,
    "applicationUrl": "https://localhost:5007;http://localhost:5006",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
```

1. Start the `Northwind.BlazorWasm.Server` project. Remember that this is an ASP.NET Core website that hosts the Blazor WebAssembly client app project.
2. Note that the app has the same functionality as before. The Blazor WebAssembly component code is executing inside the browser instead of on the server. The weather service is running on the web server. The Blazor WebAssembly app makes HTTP requests each time it wants weather forecasts.
3. Close Chrome and shut down the web server.

Similarities between Blazor Server and Blazor WebAssembly projects

The following `.razor` files are identical to those in a Blazor Server project:

- App.razor
- Shared\MainLayout.razor
- Shared\NavMenu.razor
- Shared\SurveyPrompt.razor
- Pages\Counter.razor
- Pages\Index.razor

Building components using Blazor Server

In this section, we will build a component to list, create, and edit customers in the Northwind database.

We will build it over several steps:

1. Make a Blazor Server component that renders the name of a country set as a parameter.
2. Make it work as a routable page as well as a component.
3. Implement the functionality to perform CRUD operations on customers in a database.
4. Refactor the component to work with both Blazor Server and Blazor WebAssembly.

Defining and testing a simple Blazor Server component

We will add the new component to the existing Blazor Server project:

1. In the `Northwind.BlazorServer` project (*not* the `Northwind.BlazorWasm.Server` project), in the `Pages` folder, add a new file named `Customers.razor`. In Visual Studio, the project item template is named **Razor Component**.

Good Practice: Component filenames must start with an uppercase letter, or you will have compile errors!

2. Add statements to output a heading for the `Customers` component and define a code block that defines a property to store the name of a country, as shown highlighted in the following markup:

```
<h3>Customers @(string.IsNullOrWhiteSpace(Country)
    ? "Worldwide" : "in " + Country)</h3>
@code {
  [Parameter]
  public string? Country { get; set; }
}
```

1. In the `Pages` folder, in `Index.razor`, add statements to the bottom of the file to instantiate the `Customers` component twice, once setting `Germany` as the `Country` parameter, and once without setting the country, as shown in the following markup:

```
<Customers Country="Germany" />
<Customers />
```

1. Start the `Northwind.BlazorServer` project.

2. Start Chrome and navigate to `https://localhost:5005/` and note the `Customers` components, as shown in *Figure 16.4*:

Figure 16.4: The Customers component with the Country parameter set to Germany and not set

3. Close Chrome and shut down the web server.

Making the component a routable page component

It is simple to turn this component into a routable page component with a route parameter for the country:

1. In the `Pages` folder, in the `Customers.razor` component, add a statement at the top of the file to register `/customers` as its route with an optional country route parameter, as shown in the following markup:

```
@page "/customers/{country?}"
```

1. In the `Shared` folder, in `NavMenu.razor`, at the bottom of the existing list item elements, add two list item elements for our routable page component to show customers worldwide and in Germany that both use an

icon of people, as shown in the following markup:

```
<div class="nav-item px-3">
  <NavLink class="nav-link" href="customers" Match="NavLinkMatch.All">
    <span class="oi oi-people" aria-hidden="true"></span>
    Customers Worldwide
  </NavLink>
</div>
<div class="nav-item px-3">
  <NavLink class="nav-link" href="customers/Germany">
    <span class="oi oi-people" aria-hidden="true"></span>
    Customers in Germany
  </NavLink>
</div>
```

We used an icon of people for the customers menu item. You can see the other available icons at the following link: https://iconify.design/icon-sets/oi/

1. Start the `Northwind.BlazorServer` project.
2. Start Chrome and navigate to `https://localhost:5005/`.
3. In the left navigation menu, click **Customers in Germany**, and note that the country name is correctly passed to the page component and that the component uses the same shared layout as the other page components, like `Index.razor`. Also note the URL: `https://localhost:5005/customers/Germany`
4. Close Chrome and shut down the web server.

Getting entities into a component

Now that you have seen the minimum implementation of a component, we can add some useful functionality to it. In this case, we will use the Northwind database context to fetch customers from the database:

1. In `Northwind.BlazorServer.csproj`, add a reference to the Northwind database context project for either SQL Server or SQLite, as shown in the following markup:

```
<ItemGroup>
  <!-- change Sqlite to SqlServer if you prefer -->
  <ProjectReference Include="..\Northwind.Common.DataContext.Sqlite
\Northwind.Common.DataContext.Sqlite.csproj" />
</ItemGroup>
```

1. Build the `Northwind.BlazorServer` project.
2. In `Program.cs`, import the namespace for working with the Northwind database context extension method, as shown in the following code:

```
using Packt.Shared; // AddNorthwindContext extension method
```

1. In `Program.cs`, before the call to `Build`, add a statement to register the Northwind database context in the dependency services collection, as shown in the following code:

```
builder.Services.AddNorthwindContext();
```

1. In the project folder, in `_Imports.razor`, import the namespace for working with the Northwind entities so that Blazor components that we build do not need to import the namespaces individually, as shown in the following markup:

```
@using Packt.Shared  @* Northwind entities *@
```

The `_Imports.razor` file only applies to `.razor` files. If you use code-behind `.cs` files to implement component code, then they must have namespaces imported separately or use global usings to implicitly import the namespace.

1. In the `Pages` folder, in `Customers.razor`, add statements to inject the Northwind database context and then use it to output a table of all customers, as shown in the following code:

```
@using Microsoft.EntityFrameworkCore  @* ToListAsync extension method *@
@page "/customers/{country?}"
@inject NorthwindContext db
<h3>Customers @(string.IsNullOrWhiteSpace(Country)
    ? "Worldwide" : "in " + Country)</h3>
@if (customers is null)
{
<p><em>Loading...</em></p>
}
else
{
<table class="table">
  <thead>
    <tr>
      <th>Id</th>
      <th>Company Name</th>
      <th>Address</th>
      <th>Phone</th>
      <th></th>
    </tr>
  </thead>
  <tbody>
  @foreach (Customer c in customers)
  {
    <tr>
      <td>@c.CustomerId</td>
      <td>@c.CompanyName</td>
      <td>
        @c.Address<br/>
        @c.City<br/>
        @c.PostalCode<br/>
        @c.Country
      </td>
      <td>@c.Phone</td>
      <td>
        <a class="btn btn-info" href="editcustomer/@c.CustomerId">
          <i class="oi oi-pencil"></i></a>
        <a class="btn btn-danger"
           href="deletecustomer/@c.CustomerId">
          <i class="oi oi-trash"></i></a>
      </td>
    </tr>
  }
  </tbody>
</table>
}
@code {
  [Parameter]
  public string? Country { get; set; }
  private IEnumerable<Customer>? customers;
  protected override async Task OnParametersSetAsync()
  {
    if (string.IsNullOrWhiteSpace(Country))
    {
      customers = await db.Customers.ToListAsync();
    }
    else
    {
      customers = await db.Customers
        .Where(c => c.Country == Country).ToListAsync();
    }
  }
}
```

1. Start the `Northwind.BlazorServer` project.
2. Start Chrome and navigate to `https://localhost:5005/`.

3. In the left navigation menu, click **Customers in Germany**, and note that the table of customers loads from the database and renders in the web page, as shown in *Figure 16.5*:

Figure 16.5: The list of customers in Germany

4. In the browser address bar, change `Germany` to `UK`, and note that the table of customers is filtered to only show UK customers.
5. In the left navigation menu, click **Customers Worldwide**, and note that the table of customers is unfiltered by country.
6. In the left navigation menu, click **Home**, and note that the customers component also works correctly when used as an embedded component on a page.
7. Click any of the edit or delete buttons and note that they return a message saying `Sorry, there's nothing at this address` because we have not yet implemented that functionality, and note the link, for example, to edit a customer identified by its five-character `Id`: `https://localhost:5005/editcustomer/ALFKI`
8. Close Chrome and shut down the web server.

Abstracting a service for a Blazor component

Currently, the Blazor component directly calls the Northwind database context to fetch the customers. This works fine in Blazor Server since the component executes on the server. But this component would not work when hosted in Blazor WebAssembly.

We will now create a local dependency service to enable better reuse of the components:

1. In the `Northwind.BlazorServer` project, in the `Data` folder, add a new file named `INorthwindService.cs`. The Visual Studio project item template is named **Interface**.
2. In `INorthwindService.cs`, define a contract for a local service that abstracts CRUD operations, as shown in the following code:

```
namespace Packt.Shared;
public interface INorthwindService
```

```
{
  Task<List<Customer>> GetCustomersAsync();
  Task<List<Customer>> GetCustomersAsync(string country);
  Task<Customer?> GetCustomerAsync(string id);
  Task<Customer> CreateCustomerAsync(Customer c);
  Task<Customer> UpdateCustomerAsync(Customer c);
  Task DeleteCustomerAsync(string id);
}
```

1. In the `Data` folder, add a new file named `NorthwindService.cs` and modify its contents to implement the `INorthwindService` interface by using the Northwind database context, as shown in the following code:

```
using Microsoft.EntityFrameworkCore;
namespace Packt.Shared;
public class NorthwindService : INorthwindService
{
  private readonly NorthwindContext db;
  public NorthwindService(NorthwindContext db)
  {
    this.db = db;
  }
  public Task<List<Customer>> GetCustomersAsync()
  {
    return db.Customers.ToListAsync();
  }
  public Task<List<Customer>> GetCustomersAsync(string country)
  {
    return db.Customers.Where(c => c.Country == country).ToListAsync();
  }
  public Task<Customer?> GetCustomerAsync(string id)
  {
    return db.Customers.FirstOrDefaultAsync
      (c => c.CustomerId == id);
  }
  public Task<Customer> CreateCustomerAsync(Customer c)
  {
    db.Customers.Add(c);
    db.SaveChangesAsync();
    return Task.FromResult(c);
  }
  public Task<Customer> UpdateCustomerAsync(Customer c)
  {
    db.Entry(c).State = EntityState.Modified;
    db.SaveChangesAsync();
    return Task.FromResult(c);
  }
  public Task DeleteCustomerAsync(string id)
  {
    Customer? customer = db.Customers.FirstOrDefaultAsync
      (c => c.CustomerId == id).Result;
    if (customer == null)
    {
      return Task.CompletedTask;
    }
    else
    {
      db.Customers.Remove(customer);
      return db.SaveChangesAsync();
    }
  }
}
```

1. In `Program.cs`, before the call to `Build`, add a statement to register `NorthwindService` as a transient service that implements the `INorthwindService` interface, as shown in the following code:

```
builder.Services.AddTransient<INorthwindService, NorthwindService>();
```

1. In the `Pages` folder, in `Customers.razor`, replace the directive to inject the Northwind database context with a directive to inject the registered Northwind service, as shown in the following code:

```
@inject INorthwindService service
```

1. In `Customers.razor`, modify the `OnParametersSetAsync` method to call the service instead of the Northwind database context, as shown highlighted in the following code:

```
protected override async Task OnParametersSetAsync()
{
  if (string.IsNullOrWhiteSpace(Country))
  {
    customers = await service.GetCustomersAsync();
  }
  else
  {
    customers = await service.GetCustomersAsync(Country);
  }
}
```

1. Start the `Northwind.BlazorServer` project and confirm that it retains the same functionality as before.
2. Close Chrome and shut down the web server.

So far, the component provides only a read-only table of customers. Now, we will extend it with full CRUD operations.

Defining forms using the EditForm component

Microsoft provides ready-made components for building forms. We will use them to provide create, edit, and delete functionality for customers.

Microsoft provides the `EditForm` component and several form elements such as `InputText` to make it easier to use forms with Blazor.

`EditForm` can have a model set to bind it to an object with properties and event handlers for custom validation, as well as recognizing standard Microsoft validation attributes on the model class, as shown in the following code:

```
<EditForm Model="@customer" OnSubmit="ExtraValidation">
  <DataAnnotationsValidator />
  <ValidationSummary />
  <InputText id="name" @bind-Value="customer.CompanyName" />
  <button type="submit">Submit</button>
</EditForm>
@code {
  private Customer customer = new();
  private void ExtraValidation()
  {
    // perform any extra validation
  }
}
```

As an alternative to a `ValidationSummary` component, you can use the `ValidationMessage` component to show a message next to an individual form element.

Building a shared customer detail component

Next, we will create a shared component to show the details of a customer. This will only be a component, never a page:

1. In the `Shared` folder, create a new file named `CustomerDetail.razor`. (The Visual Studio project item template is named **Razor Component**.)
2. Modify its contents to define a form to edit the properties of a customer, as shown in the following code:

```
<EditForm Model="@Customer" OnValidSubmit="@OnValidSubmit">
  <DataAnnotationsValidator />
  <div class="form-group">
```

```
    <div>
      <label>Customer Id</label>
      <div>
        <InputText @bind-Value="@Customer.CustomerId" />
        <ValidationMessage For="@(() => Customer.CustomerId)" />
      </div>
    </div>
  </div>
  <div class="form-group ">
    <div>
      <label>Company Name</label>
      <div>
        <InputText @bind-Value="@Customer.CompanyName" />
        <ValidationMessage For="@(() => Customer.CompanyName)" />
      </div>
    </div>
  </div>
  <div class="form-group ">
    <div>
      <label>Address</label>
      <div>
        <InputText @bind-Value="@Customer.Address" />
        <ValidationMessage For="@(() => Customer.Address)" />
      </div>
    </div>
  </div>
  <div class="form-group ">
    <div>
      <label>Country</label>
      <div>
        <InputText @bind-Value="@Customer.Country" />
        <ValidationMessage For="@(() => Customer.Country)" />
      </div>
    </div>
  </div>
  <button type="submit" class="btn btn-@ButtonStyle">
    @ButtonText
  </button>
</EditForm>
@code {
  [Parameter]
  public Customer Customer { get; set; } = null!;
  [Parameter]
  public string ButtonText { get; set; } = "Save Changes";
  [Parameter]
  public string ButtonStyle { get; set; } = "info";
  [Parameter]
  public EventCallback OnValidSubmit { get; set; }
}
```

Building customer create, edit, and delete components

Now we can create three routable page components that use the shared component:

1. In the `Pages` folder, create a new file named `CreateCustomer.razor`.
2. In `CreateCustomer.razor`, modify its contents to use the customer detail component to create a new customer, as shown in the following code:

```
@page "/createcustomer"
@inject INorthwindService service
@inject NavigationManager navigation
<h3>Create Customer</h3>
<CustomerDetail ButtonText="Create Customer"
                Customer="@customer"
                OnValidSubmit="@Create" />
@code {
  private Customer customer = new();
  private async Task Create()
  {
```

```
    await service.CreateCustomerAsync(customer);
    navigation.NavigateTo("customers");
  }
}
```

1. In the `Pages` folder, in `Customers.razor`, after the `<h3>` element, add a `<div>` element with a button to navigate to the create customer page component, as shown in the following markup:

```
<div class="form-group">
  <a class="btn btn-info" href="createcustomer">
  <i class="oi oi-plus"></i> Create New</a>
</div>
```

1. In the `Pages` folder, create a new file named `EditCustomer.razor` and modify its contents to use the customer detail component to edit and save changes to an existing customer, as shown in the following code:

```
@page "/editcustomer/{customerid}"
@inject INorthwindService service
@inject NavigationManager navigation
<h3>Edit Customer</h3>
<CustomerDetail ButtonText="Update"
                Customer="@customer"
                OnValidSubmit="@Update" />
@code {
  [Parameter]
  public string CustomerId { get; set; } = null!;
  private Customer? customer = new();
  protected async override Task OnParametersSetAsync()
  {
    customer = await service.GetCustomerAsync(CustomerId);
  }
  private async Task Update()
  {
    if (customer is not null)
    {
      await service.UpdateCustomerAsync(customer);
    }
    navigation.NavigateTo("customers");
  }
}
```

1. In the `Pages` folder, create a new file named `DeleteCustomer.razor` and modify its contents to use the customer detail component to show the customer that is about to be deleted, as shown in the following code:

```
@page "/deletecustomer/{customerid}"
@inject INorthwindService service
@inject NavigationManager navigation
<h3>Delete Customer</h3>
<div class="alert alert-danger">
  Warning! This action cannot be undone!
</div>
<CustomerDetail ButtonText="Delete Customer"
                ButtonStyle="danger"
                Customer="@customer"
                OnValidSubmit="@Delete" />
@code {
  [Parameter]
  public string CustomerId { get; set; } = null!;
  private Customer? customer = new();
  protected async override Task OnParametersSetAsync()
  {
    customer = await service.GetCustomerAsync(CustomerId);
  }
  private async Task Delete()
  {
    if (customer is not null)
    {
      await service.DeleteCustomerAsync(CustomerId);
    }
```

```
    navigation.NavigateTo("customers");
  }
}
```

Testing the customer components

Now we can test the customer components and how to use them to create, edit, and delete customers:

1. Start the `Northwind.BlazorServer` project.
2. Start Chrome and navigate to `https://localhost:5005/`.
3. Navigate to **Customers Worldwide** and click the **+ Create New** button.

4. Enter an invalid **Customer Id** like `ABCDEF`**,** leave the textbox, and note the validation message, as shown in *Figure 16.6*:

Figure 16.6: Creating a new customer and entering an invalid customer ID

5. Change the **Customer Id** to `ABCDE`, enter values for the other textboxes like `Alpha Corp`, `Main Street`, and `USA`, and then click the **Create Customer** button.
6. When the list of customers appears, scroll down to the bottom of the page to see the new customer.
7. On the **ABCDE** customer row, click the **Edit** icon button, change the address to something like `Upper Avenue`, click the **Update** button, and note that the customer record has been updated.
8. On the **ABCDE** customer row, click the **Delete** icon button, note the warning, click the **Delete Customer** button, and note that the customer record has been deleted.
9. Close Chrome and shut down the web server.

Building components using Blazor WebAssembly

Now we will reuse the same functionality in the Blazor WebAssembly project so that you can clearly see the key differences.

Since we abstracted the local dependency service in the `INorthwindService` interface, we will be able to reuse all the components and that interface, as well as the entity model classes. The only part that will need to be rewritten is the implementation of the `NorthwindService` class. Instead of directly calling the `NorthwindContext` class, it will call a customer Web API controller on the server side, as shown in *Figure 16.7*:

Figure 16.7: Comparing implementations using Blazor Server and Blazor WebAssembly

Configuring the server for Blazor WebAssembly

First, we need a web service that the client app can call to get and manage customers. If you completed *Chapter 15*, *Building and Consuming Web Services*, then you have a customer service in the `Northwind.WebApi` service project that you could use. However, to keep this chapter more self-contained, let's build a customer Web API controller in the `Northwind.BlazorWasm.Server` project:

Warning! Unlike previous projects, relative path references for shared projects like the entity models and the database are two levels up, for example, `"..\.."`, because we have an additional layer of folders for `Server`, `Client`, and `Shared`.

1. In the `Server` project/folder, open `Northwind.BlazorWasm.Server.csproj` and add statements to reference the Northwind database context project for either SQL Server or SQLite, as shown in the following markup:

```
<ItemGroup>
  <!-- change Sqlite to SqlServer if you prefer -->
  <ProjectReference Include="..\..\Northwind.Common.DataContext.Sqlite
\Northwind.Common.DataContext.Sqlite.csproj" />
</ItemGroup>
```

1. Build the `Northwind.BlazorWasm.Server` project.
2. In the `Server` project/folder, in `Program.cs`, add a statement to import the namespace for working with the Northwind database context extension method, as shown in the following code:

```
using Packt.Shared; // AddNorthwindContext extension method
```

1. in `Program.cs`, before the call to `Build`, add a statement to register the Northwind database context for either SQL Server or SQLite, as shown in the following code:

```
// if using SQL Server
builder.Services.AddNorthwindContext();
// if using SQLite
builder.Services.AddNorthwindContext(
  relativePath: Path.Combine("..", ".."));
```

1. In the `Server` project, in the `Controllers` folder, create a file named `CustomersController.cs` and add statements to define a Web API controller class with similar CRUD methods as before, as shown in the following code:

```
using Microsoft.AspNetCore.Mvc; // [ApiController], [Route]
using Microsoft.EntityFrameworkCore; // ToListAsync, FirstOrDefaultAsync
using Packt.Shared; // NorthwindContext, Customer
namespace Northwind.BlazorWasm.Server.Controllers;
[ApiController]
[Route("api/[controller]")]
public class CustomersController : ControllerBase
{
  private readonly NorthwindContext db;
  public CustomersController(NorthwindContext db)
  {
    this.db = db;
  }
  [HttpGet]
  public async Task<List<Customer>> GetCustomersAsync()
  {
    return await db.Customers.ToListAsync();
  }
  [HttpGet("in/{country}")] // different path to disambiguate
  public async Task<List<Customer>> GetCustomersAsync(string country)
  {
    return await db.Customers
      .Where(c => c.Country == country).ToListAsync();
  }
  [HttpGet("{id}")]
  public async Task<Customer?> GetCustomerAsync(string id)
  {
    return await db.Customers
      .FirstOrDefaultAsync(c => c.CustomerId == id);
  }
  [HttpPost]
  public async Task<Customer?> CreateCustomerAsync
    (Customer customerToAdd)
  {
    Customer? existing = await db.Customers.FirstOrDefaultAsync
      (c => c.CustomerId == customerToAdd.CustomerId);
    if (existing == null)
    {
      db.Customers.Add(customerToAdd);
      int affected = await db.SaveChangesAsync();
      if (affected == 1)
      {
        return customerToAdd;
      }
    }
    return existing;
  }
  [HttpPut]
  public async Task<Customer?> UpdateCustomerAsync(Customer c)
  {
    db.Entry(c).State = EntityState.Modified;
    int affected = await db.SaveChangesAsync();
    if (affected == 1)
    {
      return c;
    }
    return null;
  }
  [HttpDelete("{id}")]
  public async Task<int> DeleteCustomerAsync(string id)
```

```
  {
    Customer? c = await db.Customers.FirstOrDefaultAsync
      (c => c.CustomerId == id);
    if (c != null)
    {
      db.Customers.Remove(c);
      int affected = await db.SaveChangesAsync();
      return affected;
    }
    return 0;
  }
}
```

Configuring the client for Blazor WebAssembly

Now, we can reuse the components from the Blazor Server project. Since the components will be identical, we can copy them and only need to make changes to the local implementation of the abstracted Northwind service:

1. In the `Client` project, in `Northwind.BlazorWasm.Client.csproj`, add statements to reference the Northwind entity models library project (not the database context project) for either SQL Server or SQLite, as shown in the following markup:

```
<ItemGroup>
  <!-- change Sqlite to SqlServer if you prefer -->
  <ProjectReference Include="..\..\Northwind.Common.EntityModels.Sqlite\
Northwind.Common.EntityModels.Sqlite.csproj" />
</ItemGroup>
```

1. Build the `Northwind.BlazorWasm.Client` project.
2. In the `Client` project, in `_Imports.razor`, import the `Packt.Shared` namespace to make the Northwind entity model types like `Customer` and `Order` available in all Blazor components, as shown in the following code:

```
@using Packt.Shared @* Customer, Order, and so on *@
```

1. In the `Client` project, in the `Shared` folder, in `NavMenu.razor`, add a `NavLink` element for customers worldwide and in France, as shown in the following markup:

```
<div class="nav-item px-3">
  <NavLink class="nav-link" href="customers" Match="NavLinkMatch.All">
    <span class="oi oi-people" aria-hidden="true"></span>
    Customers Worldwide
  </NavLink>
</div>
<div class="nav-item px-3">
  <NavLink class="nav-link" href="customers/France">
    <span class="oi oi-people" aria-hidden="true"></span>
    Customers in France
  </NavLink>
</div>
```

1. Copy the `CustomerDetail.razor` component from the `Northwind.BlazorServer` project's `Shared` folder to the `Northwind.BlazorWasm Client` project's `Shared` folder.
2. Copy the following routable page components from the `Northwind.BlazorServer` project's `Pages` folder to the `Northwind.BlazorWasm Client` project's `Pages` folder:
   - CreateCustomer.razor
   - Customers.razor
   - DeleteCustomer.razor
   - EditCustomer.razor
3. In the `Client` project, create a `Data` folder.
4. Copy the `INorthwindService.cs` file from the `Northwind.BlazorServer` project's `Data` folder into the `Client` project's `Data` folder.

5. In the `Data` folder, add a new file named `NorthwindService.cs`, and modify its contents to implement the `INorthwindService` interface by using an `HttpClient` to call the customers Web API service, as shown in the following code:

```
using System.Net.Http.Json; // GetFromJsonAsync, ReadFromJsonAsync
using Packt.Shared; // Customer
namespace Northwind.BlazorWasm.Client.Data;
public class NorthwindService : INorthwindService
{
  private readonly HttpClient http;
  public NorthwindService(HttpClient http)
  {
    this.http = http;
  }
  public Task<List<Customer>> GetCustomersAsync()
  {
    return http.GetFromJsonAsync
      <List<Customer>>("api/customers")!;
  }
  public Task<List<Customer>> GetCustomersAsync(string country)
  {
    return http.GetFromJsonAsync
      <List<Customer>>($"api/customers/in/{country}")!;
  }
  public Task<Customer?> GetCustomerAsync(string id)
  {
    return http.GetFromJsonAsync
      <Customer>($"api/customers/{id}");
  }
  public async Task<Customer>
    CreateCustomerAsync (Customer c)
  {
    HttpResponseMessage response = await
      http.PostAsJsonAsync("api/customers", c);
    return (await response.Content
      .ReadFromJsonAsync<Customer>())!;
  }
  public async Task<Customer> UpdateCustomerAsync(Customer c)
  {
    HttpResponseMessage response = await
      http.PutAsJsonAsync("api/customers", c);
    return (await response.Content
      .ReadFromJsonAsync<Customer>())!;
  }
  public async Task DeleteCustomerAsync(string id)
  {
    HttpResponseMessage response = await
      http.DeleteAsync($"api/customers/{id}");
  }
}
```

1. In `Program.cs`, import the `Packt.Shared` and `Northwind.BlazorWasm.Client.Data` namespaces.
2. In `Program.cs`, before the call to `Build`, add a statement to register the Northwind dependency service, as shown in the following code:

```
builder.Services.AddTransient<INorthwindService, NorthwindService>();
```

Testing the Blazor WebAssembly components and service

Now we can start the Blazor WebAssembly server hosting project to test if the components work with the abstracted Northwind service that calls the customers Web API service:

1. In the `Server` project/folder, start the `Northwind.BlazorWasm.Server` website project.
2. Start Chrome, show **Developer Tools**, and select the **Network** tab.
3. Navigate to `https://localhost:5007/`.

4. Select the **Console** tab and note that Blazor WebAssembly has loaded .NET assemblies into the browser cache and that they take about 10.65 MB of space, as shown in *Figure 16.8*:

Figure 16.8: Blazor WebAssembly loading .NET assemblies into the browser cache

5. Select the **Network** tab.

6. In the left navigation menu, click **Customers Worldwide** and note the HTTP `GET` request `customers` with the JSON response containing all customers, as shown in *Figure 16.9*:

Figure 16.9: The HTTP GET request with the JSON response containing all customers

7. Click the **+ Create New** button, complete the form and click **Add Customer** to add a new customer as before, and note the HTTP `POST` request made, as shown in *Figure 16.10*:

Figure 16.10: The HTTP POST request for creating a new customer

8. Repeat the steps as before to edit and then delete the newly created customer, and note the requests in the **Network** list.
9. Close Chrome and shut down the web server.

Improving Blazor WebAssembly apps

There are common ways to improve Blazor WebAssembly apps. We'll look at some of the most popular ones now.

Enabling Blazor WebAssembly AOT

By default, the .NET runtime used by Blazor WebAssembly is doing IL interpretation using an interpreter written in WebAssembly. Unlike other .NET apps, it does not use a just-in-time (JIT) compiler, so the performance of CPU-intensive workloads is lower than you might hope for.

In .NET 6 and later, Microsoft added support for **ahead-of-time** (**AOT**) compilation, but you must explicitly opt-in because although it can dramatically improve runtime performance, AOT compilation can take several minutes on small projects like the ones in this book and potentially much longer for larger projects. The size of the compiled app is also larger than without AOT—typically twice the size. The decision to use AOT is therefore based on a balance of increased compile and browser download times, with potentially much faster runtimes.

AOT was the top requested feature in a Microsoft survey, and the lack of AOT was cited as a primary reason why some developers had not yet adopted .NET for developing **single-page applications** (**SPAs**).

Let's install the additional required workload for Blazor AOT named **.NET WebAssembly build tools** and then enable AOT for our Blazor WebAssembly project:

1. In the command prompt or terminal with admin rights, install the Blazor AOT workload, as shown in the following command:

```
dotnet workload install wasm-tools
```

1. Note the messages, as shown in the following partial output:

```
...
Installing pack Microsoft.NET.Runtime.MonoAOTCompiler.Task version 7.0.0...
Installing pack Microsoft.NETCore.App.Runtime.AOT.Cross.browser-wasm version 7.0.0...
Successfully installed workload(s) wasm-tools.
```

1. Modify the `Northwind.BlazorWasm.Client` project file to enable AOT, as shown highlighted in the following markup:

```
<PropertyGroup>
  <TargetFramework>net7.0</TargetFramework>
  ...
  <RunAOTCompilation>true</RunAOTCompilation>
</PropertyGroup>
```

1. Publish the `Northwind.BlazorWasm.Client` project, as shown in the following command:

```
dotnet publish -c Release
```

Be patient. Wait for the process to finish. The process can take at least 20 minutes even on a modern multi-core CPU.

1. Note that more than 70 assemblies have AOT applied, as shown in the following partial output:

```
Northwind.BlazorWasm.Shared -> C:\cs11dotnet7\PracticalApps\Northwind.BlazorWasm\Shared\bin\Re
Northwind.Common.EntityModels.Sqlite -> C:\cs11dotnet7\PracticalApps\Northwind.Common.EntityMo
Northwind.BlazorWasm.Client -> C:\cs11dotnet7\PracticalApps\Northwind.BlazorWasm\Client\bin\Re
Northwind.BlazorWasm.Client (Blazor output) -> C:\cs11dotnet7\PracticalApps\Northwind.BlazorWa
Optimizing assemblies for size may change the behavior of the app. Be sure to test after publi
Optimizing assemblies for size. This process might take a while.
Northwind.Common.EntityModels.Sqlite -> C:\cs11dotnet7\PracticalApps\Northwind.Common.EntityMo
Northwind.BlazorWasm.Shared -> C:\cs11dotnet7\PracticalApps\Northwind.BlazorWasm\Shared\bin\Re
Optimizing assemblies for size may change the behavior of the app. Be sure to test after publi
AOT'ing 73 assemblies
[1/72] System.ComponentModel.dll -> System.ComponentModel.dll.bc
...
[72/72] Microsoft.EntityFrameworkCore.Abstractions.dll -> Microsoft.EntityFrameworkCore.Abstra
Compiling native assets with emcc with -Oz. This may take a while ...
[1/3] pinvoke.c -> pinvoke.o [took 0.78s]
[2/3] corebindings.c -> corebindings.o [took 0.98s]
[3/3] driver.c -> driver.o [took 1.15s]
Compiling assembly bitcode files with -O2 ...
[1/72] System.ComponentModel.dll.bc -> System.ComponentModel.dll.o [took 2.61s]
...
Linking with emcc. This may take a while ...
...
```

```
Optimizing dotnet.wasm ...
Compressing Blazor WebAssembly publish artifacts. This may take a while...
```

1. Navigate to the `Northwind.BlazorWasm\Client\bin\release\net7.0\publish` folder and note the increased size of the downloaded files.

Without AOT, the downloaded Blazor WebAssembly app took about 10.65 MB of space. With AOT, it took about 95.9 MB. This increase in size will affect a website visitor's experience.

Good Practice: The use of AOT is a balance between larger initial download (and therefore slower start) and faster potential execution after that. Depending on the specifics of your app, AOT might not be worth it.

Exploring Progressive Web App support

Progressive Web App (PWA) support in Blazor WebAssembly projects means that the web app gains the following benefits:

- It acts as a normal web page until the visitor explicitly decides to progress to a full app experience.
- After the app is installed, launch it from the OS's start menu or desktop.
- It visually appears in its own app window instead of a browser tab.
- It works offline.
- It automatically updates.

Let us see PWA support in action:

1. Start the `Northwind.BlazorWasm.Server` project.
2. Start Chrome and navigate to `https://localhost:5007/`.

3. In Chrome, in the address bar on the right, click the icon with the tooltip **Install Northwind.BlazorWasm**, as shown in *Figure 16.11*:

Figure 16.11: Installing Northwind.BlazorWasm as an app

4. Click the **Install** button.
5. Close the app window and close Chrome.
6. Launch the **Northwind.BlazorWasm** app from your Windows Start menu or macOS Launchpad and note that it has a full app experience. Search for `northwind` if it does not appear in the recent apps list.
7. On the right of the title bar, click the three dots menu and note that you can uninstall the app, but do not do so yet.
8. Navigate to **Developer Tools**. On Windows, you can press F12 or Ctrl + Shift + I. On macOS, you can press Cmd + Shift + I.
9. Select the **Network** tab and then, in the **Throttling** dropdown, select the **Offline** preset.

10. In the left navigation menu, click **Home** and then click **Customers Worldwide**, and note the failure to load any customers and the error message at the bottom of the app window, as shown in *Figure 16.12*:

Figure 16.12: Failure to load any customers when the network is offline

11. In **Developer Tools**, set **Throttling** back to **Disabled: No throttling**.
12. Click the **Reload** link in the yellow error bar at the bottom of the app and note that functionality returns.
13. You could now uninstall the PWA app or just close it.

Implementing offline support for PWAs

We could improve the experience by caching HTTP `GET` responses from the Web API service locally, storing new, modified, or deleted customers locally, and then synchronizing with the server later by making the stored HTTP requests once network connectivity is restored. But that takes a lot of effort to implement well, so it is beyond the scope of this book.

Understanding the browser compatibility analyzer for Blazor WebAssembly

With .NET 6 and later, Microsoft has unified the .NET library for all workloads. However, although in theory, this means that a Blazor WebAssembly app has full access to all .NET APIs, in practice, it runs inside a browser sandbox so there are limitations. If you call an unsupported API, this will throw a `PlatformNotSupportedException`.

To be forewarned about unsupported APIs, you can add a platform compatibility analyzer that will warn you when your code uses APIs that are not supported by browsers.

Blazor WebAssembly App and **Razor Class Library** project templates automatically enable browser compatibility checks.

To manually activate browser compatibility checks, for example, in a **Class Library** project, add an entry to the project file, as shown in the following markup:

```
<ItemGroup>
  <SupportedPlatform Include="browser" />
</ItemGroup>
```

Microsoft decorates unsupported APIs, as shown in the following code:

```
[UnsupportedOSPlatform("browser")]
public void DoSomethingOutsideTheBrowserSandbox()
{
  ...
}
```

Good Practice: If you create libraries that should not be used in Blazor WebAssembly apps, then you should decorate your APIs in the same way.

Interop with JavaScript

By default, Blazor components do not have access to browser capabilities like local storage, geolocation, and media capture, or any JavaScript libraries like React or Vue. If you need to interact with them, you can use JavaScript Interop.

Let's see an example that uses the browser window's alert box and local storage that can persist up to 5 MB of data per visitor indefinitely:

1. In the `Northwind.BlazorServer` project, in the `wwwroot` folder, add a folder named `scripts`.
2. In the `scripts` folder, add a file named `interop.js`, and then modify its contents, as shown in the following code:

```
function messageBox(message) {
  window.alert(message);
}
function setColorInStorage() {
  if (typeof (Storage) !== "undefined") {
    localStorage.setItem("color",
      document.getElementById("colorBox").value);
  }
}
function getColorFromStorage() {
  if (typeof (Storage) !== "undefined") {
    document.getElementById("colorBox").value =
      localStorage.getItem("color");
  }
}
```

1. In the `Pages` folder, in `_Layout.cshtml`, after the `script` element that adds Blazor Server support, add a `script` element that references the JavaScript file that you just created, as shown in the following code:

```
<script src="scripts/interop.js"></script>
```

1. In the `Pages` folder, in `Index.razor`, delete the two `Customers` component instances and then add a button and a code block that uses the Blazor JavaScript runtime dependency service to call a JavaScript function, as shown in the following code:

```
<button type="button" class="btn btn-info" @onclick="AlertBrowser">
  Poke the browser</button>
<hr />
<input id="colorBox" />
<button type="button" class="btn btn-info" @onclick="SetColor">
  Set Color</button>
<button type="button" class="btn btn-info" @onclick="GetColor">
  Get Color</button>
@code {
  [Inject]
  public IJSRuntime JSRuntime { get; set; } = null!;
  public async Task AlertBrowser()
  {
    await JSRuntime.InvokeVoidAsync(
      "messageBox", "Blazor poking the browser");
  }
  public async Task SetColor()
  {
    await JSRuntime.InvokeVoidAsync("setColorInStorage");
  }
  public async Task GetColor()
  {
    await JSRuntime.InvokeVoidAsync("getColorFromStorage");
  }
}
```

1. Start the `Northwind.BlazorServer` project.
2. Start Chrome and navigate to `https://localhost:5005/`.
3. On the home page, in the textbox, enter `red` and then click the **Set Color** button.

4. Show **Developer Tools**, select the **Application** tab, expand **Local Storage**, select `https://localhost:5005`, and note the key-value pair color-red, as shown in *Figure 16.13*:

Figure 16.13: Storing a color in browser local storage using JavaScript Interop

5. Close Chrome and shut down the web server.
6. Start the `Northwind.BlazorServer` project.
7. Start Chrome and navigate to `https://localhost:5005/`.
8. On the home page, click the **Get Color** button and note that the value red is shown in the textbox, retrieved from local storage between visitor sessions.
9. Click the **Poke the browser** button and note the message that appears.
10. Close Chrome and shut down the web server.

Libraries of Blazor components

There are many libraries of Blazor components. Paid component libraries are from companies like Telerik, DevExpress, and Syncfusion. Open source Blazor component libraries include the following:

- Radzen Blazor Components: https://blazor.radzen.com/
- Awesome Open Source Blazor Projects: https://awesomeopensource.com/projects/blazor

Practicing and exploring

Test your knowledge and understanding by answering some questions, get some hands-on practice, and explore this chapter's topics with deeper research.

Exercise 16.1 – Test your knowledge

Answer the following questions:

1. What are the two primary hosting models for Blazor, and how are they different?
2. In a Blazor Server website project, compared to an ASP.NET Core MVC website project, what extra configuration is required in the Startup class?
3. One of the benefits of Blazor is being able to implement client-side components using C# and .NET instead of JavaScript. Does a Blazor component need any JavaScript?
4. In a Blazor project, what does the `App.razor` file do?
5. What is a benefit of using the `<NavLink>` component?
6. How can you pass a value into a component?
7. What is a benefit of using the `<EditForm>` component?
8. How can you execute some statements when parameters are set?
9. How can you execute some statements when a component appears?
10. What are two key differences in the `Program` class between a Blazor Server and Blazor WebAssembly project?

Exercise 16.2 – Practice by creating a times table component

In the `Northwind.BlazorServer` project, create a routable page component that renders a times table based on a parameter named `Number` and then test your component in two ways.

First, by adding an instance of your component to the `Index.razor` file, as shown in the following markup:

```
<timestable Number="6" />
```

Second, by entering a path in the browser address bar, as shown in the following link:

```
https://localhost:5005/timestable/6
```

Exercise 16.3 – Practice by creating a country navigation item

Add an action method to the `CustomersController` class to return a list of country names.

In the shared `NavMenu` component, call the customer's web service to get the list of country names and loop through them, creating a menu item for each country.

Exercise 16.4 – Explore topics

Use the links on the following page to learn more detail about the topics covered in this chapter:

https://github.com/markjprice/cs11dotnet7/blob/main/book-links.md#chapter-16---building-user-interfaces-using-blazor

Summary

In this chapter, you learned how to build Blazor components hosted for both Server and WebAssembly. You saw some of the key differences between the two hosting models, like how data should be managed using dependency services.

In the *Epilogue*, I will make some suggestions for books to take you deeper into C# and .NET.

PERCEPTIONS OF A RENEGADE MIND

DAVID ICKE

PERCEPTIONS
OF A
RENEGADE
MIND
DAVID ICKE

PERCEPTIONS OF A RENEGADE MIND

ickonic
publishing

First published in July 2021.

New Enterprise House
St Helens Street
Derby
DE1 3GY
UK

email: gareth.icke@davidicke.com

Copyright © 2021 David Icke

No part of this book may be reproduced in any form without permission from the Publisher, except for the quotation of brief passages in criticism

Cover Design: Gareth Icke
Book Design: Neil Hague

British Library Cataloguing-in Publication Data
A catalogue record for this book is available from the British Library

eISBN 978-18384153-1-0

PERCEPTIONS OF A RENEGADE MIND

DAVID ICKE

Dedication:

To *Freeeeeedom!*

ICKONIC
THE ALTERNATIVE

NEW. DIFFERENT. REVOLUTIONARY

HUNDREDS OF CUTTING EDGE DOCUMENTARIES,
FEATURE FILMS, SERIES & PODCASTS.

SIGN UP NOW AT ICKONIC.COM

THE LIFE STORY OF DAVID ICKE

RENEGADE

THE FEATURE LENGTH FILM

AVAILABLE NOW AT DAVIDICKE.COM

Renegade:
Adjective
'Having rejected tradition: Unconventional.'
Merriam-Webster Dictionary

Acquiescence to tyranny is the death of the spirit

You may be 38 years old, as I happen to be. And one day, some great opportunity stands before you and calls you to stand up for some great principle, some great issue, some great cause. And you refuse to do it because you are afraid … You refuse to do it because you want to live longer … You're afraid that you will lose your job, or you are afraid that you will be criticised or that you will lose your popularity, or you're afraid that somebody will stab you, or shoot at you or bomb your house; so you refuse to take the stand.

Well, you may go on and live until you are 90, but you're just as dead at 38 as you would be at 90. And the cessation of breathing in your life is but the belated announcement of an earlier death of the spirit.

Martin Luther King

How the few control the many and always have – the many do whatever they're told

'Forward, the Light Brigade!'
Was there a man dismayed?
Not though the soldier knew
Someone had blundered.
Theirs not to make reply,
Theirs not to reason why,
Theirs but to do and die.
Into the valley of Death
Rode the six hundred.

Cannon to right of them,
Cannon to left of them,
Cannon in front of them
Volleyed and thundered;
Stormed at with shot and shell,
Boldly they rode and well,
Into the jaws of Death,
Into the mouth of hell
Rode the six hundred

Alfred Lord Tennyson (1809-1892)

The mist is lifting slowly
I can see the way ahead
And I've left behind the empty streets
That once inspired my life
And the strength of the emotion
Is like thunder in the air
'Cos the promise that we made each other
Haunts me to the end

The secret of your beauty
And the mystery of your soul
I've been searching for in everyone I meet
And the times I've been mistaken
It's impossible to say
And the grass is growing
Underneath our feet

The words that I remember
From my childhood still are true
That there's none so blind
As those who will not see
And to those who lack the courage
And say it's dangerous to try
Well they just don't know
That love eternal will not be denied

I know you're out there somewhere
Somewhere, somewhere
I know you're out there somewhere

Somewhere you can hear my voice
I know I'll find you somehow
Somehow, somehow
I know I'll find you somehow
And somehow I'll return again to you

The Moody Blues

Are you a gutless wonder - or a Renegade Mind?

Monuments put from pen to paper,
Turns me into a gutless wonder,
And if you tolerate this,
Then your children will be next.
Gravity keeps my head down,
Or is it maybe shame ...

Manic Street Preachers

Rise like lions after slumber
In unvanquishable number.
Shake your chains to earth like dew
Which in sleep have fallen on you.
Ye are many – they are few.

Percy Shelley

Contents

CHAPTER 1 'I'm thinking' – Oh, but *are* you?
CHAPTER 2 Renegade perception
CHAPTER 3 The Pushbacker sting
CHAPTER 4 'Covid': The calculated catastrophe
CHAPTER 5 There *is no* 'virus'
CHAPTER 6 Sequence of deceit
CHAPTER 7 War on your mind
CHAPTER 8 'Reframing' insanity
CHAPTER 9 We must have it? So what is it?
CHAPTER 10 Human 2.0
CHAPTER 11 Who controls the Cult?
CHAPTER 12 Escaping Wetiko

POSTSCRIPT
APPENDIX Cowan-Kaufman-Morell Statement on Virus Isolation
BIBLIOGRAPHY
INDEX

CHAPTER ONE

I'm thinking' – Oh, but *are* you?

Think for yourself and let others enjoy the privilege of doing so too
Voltaire

French-born philosopher, mathematician and scientist René Descartes became famous for his statement in Latin in the 17th century which translates into English as: 'I think, therefore I am.'

On the face of it that is true. Thought reflects perception and perception leads to both behaviour and self-identity. In that sense 'we' are what we think. But who or what is doing the thinking and is thinking the only route to perception? Clearly, as we shall see, 'we' are not always the source of 'our' perception, indeed with regard to humanity as a whole this is rarely the case; and thinking is far from the only means of perception. Thought is the village idiot compared with other expressions of consciousness that we all have the potential to access and tap into. This has to be true when we *are* those other expressions of consciousness which are infinite in nature. We have forgotten this, or, more to the point, been manipulated to forget.

These are not just the esoteric musings of the navel. The whole foundation of human control and oppression is control of perception. Once perception is hijacked then so is behaviour which is dictated by perception. Collective perception becomes collective behaviour and collective behaviour is what we call human society. Perception is all and those behind human control know that which is

why perception is the target 24/7 of the psychopathic manipulators that I call the Global Cult. They know that if they dictate perception they will dictate behaviour and collectively dictate the nature of human society. They are further aware that perception is formed from information received and if they control the circulation of information they will to a vast extent direct human behaviour. Censorship of information and opinion has become globally Nazi-like in recent years and never more blatantly than since the illusory 'virus pandemic' was triggered out of China in 2019 and across the world in 2020. Why have billions submitted to house arrest and accepted fascistic societies in a way they would have never believed possible? Those controlling the information spewing from government, mainstream media and Silicon Valley (all controlled by the same Global Cult networks) told them they were in danger from a 'deadly virus' and only by submitting to house arrest and conceding their most basic of freedoms could they and their families be protected. This monumental and provable lie became the *perception* of the billions and therefore the *behaviour* of the billions. In those few words you have the whole structure and modus operandi of human control. Fear is a perception – **F**alse **E**motion **A**ppearing **R**eal – and fear is the currency of control. In short … get them by the balls (or give them the impression that you have) and their hearts and minds will follow. Nothing grips the dangly bits and freezes the rear-end more comprehensively than fear.

World number 1

There are two 'worlds' in what appears to be one 'world' and the prime difference between them is knowledge. First we have the mass of human society in which the population is maintained in coldly-calculated ignorance through control of information and the 'education' (indoctrination) system. That's all you really need to control to enslave billions in a perceptual delusion in which what are perceived to be *their* thoughts and opinions are ever-repeated mantras that the system has been downloading all their lives through 'education', media, science, medicine, politics and academia

in which the personnel and advocates are themselves overwhelmingly the perceptual products of the same repetition. Teachers and academics in general are processed by the same programming machine as everyone else, but unlike the great majority they never leave the 'education' program. It gripped them as students and continues to grip them as programmers of subsequent generations of students. The programmed become the programmers – the programmed programmers. The same can largely be said for scientists, doctors and politicians and not least because as the American writer Upton Sinclair said: 'It is difficult to get a man to understand something when his salary depends upon his not understanding it.' If your career and income depend on thinking the way the system demands then you will – bar a few free-minded exceptions – concede your mind to the Perceptual Mainframe that I call the Postage Stamp Consensus. This is a tiny band of perceived knowledge and possibility 'taught' (downloaded) in the schools and universities, pounded out by the mainstream media and on which all government policy is founded. Try thinking, and especially speaking and acting, outside of the 'box' of consensus and see what that does for your career in the Mainstream Everything which bullies, harasses, intimidates and ridicules the population into compliance. Here we have the simple structure which enslaves most of humanity in a perceptual prison cell for an entire lifetime and I'll go deeper into this process shortly. Most of what humanity is taught as fact is nothing more than programmed belief. American science fiction author Frank Herbert was right when he said: 'Belief can be manipulated. Only knowledge is dangerous.' In the 'Covid' age belief is promoted and knowledge is censored. It was always so, but never to the extreme of today.

World number 2

A 'number 2' is slang for 'doing a poo' and how appropriate that is when this other 'world' is doing just that on humanity every minute of every day. World number 2 is a global network of secret societies and semi-secret groups dictating the direction of society via

governments, corporations and authorities of every kind. I have spent more than 30 years uncovering and exposing this network that I call the Global Cult and knowing its agenda is what has made my books so accurate in predicting current and past events. Secret societies are secret for a reason. They want to keep their hoarded knowledge to themselves and their chosen initiates and to hide it from the population which they seek through ignorance to control and subdue. The whole foundation of the division between World 1 and World 2 is *knowledge*. What number 1 knows number 2 must not. Knowledge they have worked so hard to keep secret includes (a) the agenda to enslave humanity in a centrally-controlled global dictatorship, and (b) the nature of reality and life itself. The latter (b) must be suppressed to allow the former (a) to prevail as I shall be explaining. The way the Cult manipulates and interacts with the population can be likened to a spider's web. The 'spider' sits at the centre in the shadows and imposes its will through the web with each strand represented in World number 2 by a secret society, satanic or semi-secret group, and in World number 1 – the world of the seen – by governments, agencies of government, law enforcement, corporations, the banking system, media conglomerates and Silicon Valley (Fig 1 overleaf). The spider and the web connect and coordinate all these organisations to pursue the same global outcome while the population sees them as individual entities working randomly and independently. At the level of the web governments *are* the banking system *are* the corporations *are* the media *are* Silicon Valley *are* the World Health Organization working from their inner cores as one unit. Apparently unconnected countries, corporations, institutions, organisations and people are on the *same team* pursuing the same global outcome. Strands in the web immediately around the spider are the most secretive and exclusive secret societies and their membership is emphatically restricted to the Cult inner-circle emerging through the generations from particular bloodlines for reasons I will come to. At the core of the core you would get them in a single room. That's how many people are dictating the direction of human society and its transformation

through the 'Covid' hoax and other means. As the web expands out from the spider we meet the secret societies that many people will be aware of – the Freemasons, Knights Templar, Knights of Malta, Opus Dei, the inner sanctum of the Jesuit Order, and such like. Note how many are connected to the Church of Rome and there is a reason for that. The Roman Church was established as a revamp, a rebranding, of the relocated 'Church' of Babylon and the Cult imposing global tyranny today can be tracked back to Babylon and Sumer in what is now Iraq.

Figure 1: The global web through which the few control the many. (Image Neil Hague.)

Inner levels of the web operate in the unseen away from the public eye and then we have what I call the cusp organisations located at the point where the hidden meets the seen. They include a series of satellite organisations answering to a secret society founded in London in the late 19th century called the Round Table and among them are the Royal Institute of International Affairs (UK, founded in 1920); Council on Foreign Relations (US, 1921); Bilderberg Group (worldwide, 1954); Trilateral Commission (US/worldwide, 1972); and the Club of Rome (worldwide, 1968) which was created to exploit environmental concerns to justify the centralisation of global power to 'save the planet'. The Club of Rome instigated with others the human-caused climate change hoax which has led to all the 'green

new deals' demanding that very centralisation of control. Cusp organisations, which include endless 'think tanks' all over the world, are designed to coordinate a single global policy between political and business leaders, intelligence personnel, media organisations and anyone who can influence the direction of policy in their own sphere of operation. Major players and regular attenders will know what is happening – or some of it – while others come and go and are kept overwhelmingly in the dark about the big picture. I refer to these cusp groupings as semi-secret in that they can be publicly identified, but what goes on at the inner-core is kept very much 'in house' even from most of their members and participants through a fiercely-imposed system of compartmentalisation. Only let them know what they need to know to serve your interests and no more. The structure of secret societies serves as a perfect example of this principle. Most Freemasons never get higher than the bottom three levels of 'degree' (degree of knowledge) when there are 33 official degrees of the Scottish Rite. Initiates only qualify for the next higher 'compartment' or degree if those at that level choose to allow them. Knowledge can be carefully assigned only to those considered 'safe'. I went to my local Freemason's lodge a few years ago when they were having an 'open day' to show how cuddly they were and when I chatted to some of them I was astonished at how little the rank and file knew even about the most ubiquitous symbols they use. The mushroom technique – keep them in the dark and feed them bullshit – applies to most people in the web as well as the population as a whole. Sub-divisions of the web mirror in theme and structure transnational corporations which have a headquarters somewhere in the world dictating to all their subsidiaries in different countries. Subsidiaries operate in their methodology and branding to the same centrally-dictated plan and policy in pursuit of particular ends. The Cult web functions in the same way. Each country has its own web as a subsidiary of the global one. They consist of networks of secret societies, semi-secret groups and bloodline families and their job is to impose the will of the spider and the global web in their particular country. Subsidiary networks control and manipulate the national political system, finance, corporations, media, medicine, etc. to

ensure that they follow the globally-dictated Cult agenda. These networks were the means through which the 'Covid' hoax could be played out with almost every country responding in the same way.

The 'Yessir' pyramid

Compartmentalisation is the key to understanding how a tiny few can dictate the lives of billions when combined with a top-down sequence of imposition and acquiescence. The inner core of the Cult sits at the peak of the pyramidal hierarchy of human society (Fig 2 overleaf). It imposes its will – its agenda for the world – on the level immediately below which acquiesces to that imposition. This level then imposes the Cult will on the level below them which acquiesces and imposes on the next level. Very quickly we meet levels in the hierarchy that have no idea there even is a Cult, but the sequence of imposition and acquiescence continues down the pyramid in just the same way. 'I don't know why we are doing this but the order came from "on-high" and so we better just do it.' Alfred Lord Tennyson said of the cannon fodder levels in his poem *The Charge of the Light Brigade*: 'Theirs not to reason why; theirs but to do and die.' The next line says that 'into the valley of death rode the six hundred' and they died because they obeyed without question what their perceived 'superiors' told them to do. In the same way the population capitulated to 'Covid'. The whole hierarchical pyramid functions like this to allow the very few to direct the enormous many. Eventually imposition-acquiescence-imposition-acquiescence comes down to the mass of the population at the foot of the pyramid. If they acquiesce to those levels of the hierarchy imposing on them (governments/law enforcement/doctors/media) a circuit is completed between the population and the handful of super-psychopaths in the Cult inner core at the top of the pyramid. Without a circuit-breaking refusal to obey, the sequence of imposition and acquiescence allows a staggeringly few people to impose their will upon the entirety of humankind. We are looking at the very sequence that has subjugated billions since the start of 2020. Our freedom has not been taken from us. Humanity has given it

away. Fascists do not impose fascism because there are not enough of them. Fascism is imposed by the population acquiescing to fascism. Put another way allowing their perceptions to be programmed to the extent that leads to the population giving their freedom away by giving their perceptions – their mind – away. If this circuit is not broken by humanity ceasing to cooperate with their own enslavement then nothing can change. For that to happen people have to critically think and see through the lies and window dressing and then summon the backbone to act upon what they see. The Cult spends its days working to stop either happening and its methodology is systematic and highly detailed, but it can be overcome and that is what this book is all about.

Figure 2: The simple sequence of imposition and compliance that allows a handful of people at the peak of the pyramid to dictate the lives of billions.

The Life Program

Okay, back to world number 1 or the world of the 'masses'. Observe the process of what we call 'life' and it is a perceptual download from cradle to grave. The Cult has created a global structure in which perception can be programmed and the program continually topped-up with what appears to be constant confirmation that the program is indeed true reality. The important word here is 'appears'.

This is the structure, the fly-trap, the Postage Stamp Consensus or Perceptual Mainframe, which represents that incredibly narrow band of perceived possibility delivered by the 'education' system, mainstream media, science and medicine. From the earliest age the download begins with parents who have themselves succumbed to the very programming their children are about to go through. Most parents don't do this out of malevolence and mostly it is quite the opposite. They do what they believe is best for their children and that is what the program has told them is best. Within three or four years comes the major transition from parental programming to full-blown state (Cult) programming in school, college and university where perceptually-programmed teachers and academics pass on their programming to the next generations. Teachers who resist are soon marginalised and their careers ended while children who resist are called a problem child for whom Ritalin may need to be prescribed. A few years after entering the 'world' children are under the control of authority figures representing the state telling them when they have to be there, when they can leave and when they can speak, eat, even go to the toilet. This is calculated preparation for a lifetime of obeying authority in all its forms. Reflex-action fear of authority is instilled by authority from the start. Children soon learn the carrot and stick consequences of obeying or defying authority which is underpinned daily for the rest of their life. Fortunately I daydreamed through this crap and never obeyed authority simply because it told me to. This approach to my alleged 'betters' continues to this day. There can be consequences of pursuing open-minded freedom in a world of closed-minded conformity. I spent a lot of time in school corridors after being ejected from the classroom for not taking some of it seriously and now I spend a lot of time being ejected from Facebook, YouTube and Twitter. But I can tell you that being true to yourself and not compromising your self-respect is far more exhilarating than bowing to authority for authority's sake. You don't have to be a sheep to the shepherd (authority) and the sheep dog (fear of not obeying authority).

The perceptual download continues throughout the formative years in school, college and university while script-reading 'teachers', 'academics' 'scientists', 'doctors' and 'journalists' insist that ongoing generations must be as programmed as they are. Accept the program or you will not pass your 'exams' which confirm your 'degree' of programming. It is tragic to think that many parents pressure their offspring to work hard at school to download the program and qualify for the next stage at college and university. The late, great, American comedian George Carlin said: 'Here's a bumper sticker I'd like to see: We are proud parents of a child who has resisted his teachers' attempts to break his spirit and bend him to the will of his corporate masters.' Well, the best of luck finding many of those, George. Then comes the moment to leave the formal programming years in academia and enter the 'adult' world of work. There you meet others in your chosen or prescribed arena who went through the same Postage Stamp Consensus program before you did. There is therefore overwhelming agreement between almost everyone on the basic foundations of Postage Stamp reality and the rejection, even contempt, of the few who have a mind of their own and are prepared to use it. This has two major effects. Firstly, the consensus confirms to the programmed that their download is really how things are. I mean, everyone knows that, right? Secondly, the arrogance and ignorance of Postage Stamp adherents ensure that anyone questioning the program will have unpleasant consequences for seeking their own truth and not picking their perceptions from the shelf marked: 'Things you must believe without question and if you don't you're a dangerous lunatic conspiracy theorist and a harebrained nutter'.

Every government, agency and corporation is founded on the same Postage Stamp prison cell and you can see why so many people believe the same thing while calling it their own 'opinion'. Fusion of governments and corporations in pursuit of the same agenda was the definition of fascism described by Italian dictator Benito Mussolini. The pressure to conform to perceptual norms downloaded for a lifetime is incessant and infiltrates society right

down to family groups that become censors and condemners of their own 'black sheep' for not, ironically, being sheep. We have seen an explosion of that in the 'Covid' era. Cult-owned global media unleashes its propaganda all day every day in support of the Postage Stamp and targets with abuse and ridicule anyone in the public eye who won't bend their mind to the will of the tyranny. Any response to this is denied (certainly in my case). They don't want to give a platform to expose official lies. Cult-owned-and-created Internet giants like Facebook, Google, YouTube and Twitter delete you for having an unapproved opinion. Facebook boasts that its AI censors delete 97-percent of 'hate speech' before anyone even reports it. Much of that 'hate speech' will simply be an opinion that Facebook and its masters don't want people to see. Such perceptual oppression is widely known as fascism. Even Facebook executive Benny Thomas, a 'CEO Global Planning Lead', said in comments secretly recorded by investigative journalism operation Project Veritas that Facebook is 'too powerful' and should be broken up:

> I mean, no king in history has been the ruler of two billion people, but Mark Zuckerberg is … And he's 36. That's too much for a 36-year-old ... You should not have power over two billion people. I just think that's wrong.

Thomas said Facebook-owned platforms like Instagram, Oculus, and WhatsApp needed to be separate companies. 'It's too much power when they're all one together'. That's the way the Cult likes it, however. We have an executive of a Cult organisation in Benny Thomas that doesn't know there is a Cult such is the compartmentalisation. Thomas said that Facebook and Google 'are no longer companies, they're countries'. Actually they are more powerful than countries on the basis that if you control information you control perception and control human society.

I love my oppressor

Another expression of this psychological trickery is for those who realise they are being pressured into compliance to eventually

convince themselves to believe the official narratives to protect their self-respect from accepting the truth that they have succumbed to meek and subservient compliance. Such people become some of the most vehement defenders of the system. You can see them everywhere screaming abuse at those who prefer to think for themselves and by doing so reminding the compliers of their own capitulation to conformity. 'You are talking dangerous nonsense you Covidiot!!' Are you trying to convince me or yourself? It is a potent form of Stockholm syndrome which is defined as: 'A psychological condition that occurs when a victim of abuse identifies and attaches, or bonds, positively with their abuser.' An example is hostages bonding and even 'falling in love' with their kidnappers. The syndrome has been observed in domestic violence, abused children, concentration camp inmates, prisoners of war and many and various Satanic cults. These are some traits of Stockholm syndrome listed at goodtherapy.org:

- Positive regard towards perpetrators of abuse or captor [see 'Covid'].
- Failure to cooperate with police and other government authorities when it comes to holding perpetrators of abuse or kidnapping accountable [or in the case of 'Covid' cooperating with the police to enforce and defend their captors' demands].
- Little or no effort to escape [see 'Covid'].
- Belief in the goodness of the perpetrators or kidnappers [see 'Covid'].
- Appeasement of captors. This is a manipulative strategy for maintaining one's safety. As victims get rewarded – perhaps with less abuse or even with life itself – their appeasing behaviours are reinforced [see 'Covid'].
- Learned helplessness. This can be akin to 'if you can't beat 'em, join 'em'. As the victims fail to escape the abuse or captivity, they may start giving up and soon realize it's just easier for everyone if they acquiesce all their power to their captors [see 'Covid'].

- Feelings of pity toward the abusers, believing they are actually victims themselves. Because of this, victims may go on a crusade or mission to 'save' [protect] their abuser [see the venom unleashed on those challenging the official 'Covid' narrative].
- Unwillingness to learn to detach from their perpetrators and heal. In essence, victims may tend to be less loyal to themselves than to their abuser [*definitely* see 'Covid'].

Ponder on those traits and compare them with the behaviour of great swathes of the global population who have defended governments and authorities which have spent every minute destroying their lives and livelihoods and those of their children and grandchildren since early 2020 with fascistic lockdowns, house arrest and employment deletion to 'protect' them from a 'deadly virus' that their abusers' perceptually created to bring about this very outcome. We are looking at mass Stockholm syndrome. All those that agree to concede their freedom will believe those perceptions are originating in their own independent 'mind' when in fact by conceding their reality to Stockholm syndrome they have by definition conceded any independence of mind. Listen to the 'opinions' of the acquiescing masses in this 'Covid' era and what gushes forth is the repetition of the official version of everything delivered unprocessed, unfiltered and unquestioned. The whole programming dynamic works this way. I must be free because I'm told that I am and so I think that I am.

You can see what I mean with the chapter theme of 'I'm thinking – Oh, but *are* you?' The great majority are not thinking, let alone for themselves. They are repeating what authority has told them to believe which allows them to be controlled. Weaving through this mentality is the fear that the 'conspiracy theorists' are right and this again explains the often hysterical abuse that ensues when you dare to contest the official narrative of anything. Denial is the mechanism of hiding from yourself what you don't want to be true. Telling people what they want to hear is easy, but it's an infinitely greater challenge to tell them what they would rather not be happening.

One is akin to pushing against an open door while the other is met with vehement resistance no matter what the scale of evidence. I don't want it to be true so I'll convince myself that it's not. Examples are everywhere from the denial that a partner is cheating despite all the signs to the reflex-action rejection of any idea that world events in which country after country act in exactly the same way are centrally coordinated. To accept the latter is to accept that a force of unspeakable evil is working to destroy your life and the lives of your children with nothing too horrific to achieve that end. Who the heck wants that to be true? But if we don't face reality the end is duly achieved and the consequences are far worse and ongoing than breaking through the walls of denial today with the courage to make a stand against tyranny.

Connect the dots – but how?

A crucial aspect of perceptual programming is to portray a world in which everything is random and almost nothing is connected to anything else. Randomness cannot be coordinated by its very nature and once you perceive events as random the idea they could be connected is waved away as the rantings of the tinfoil-hat brigade. You can't plan and coordinate random you idiot! No, you can't, but you can hide the coldly-calculated and long-planned behind the *illusion* of randomness. A foundation manifestation of the Renegade Mind is to scan reality for patterns that connect the apparently random and turn pixels and dots into pictures. This is the way I work and have done so for more than 30 years. You look for similarities in people, modus operandi and desired outcomes and slowly, then ever quicker, the picture forms. For instance: There would seem to be no connection between the 'Covid pandemic' hoax and the human-caused global-warming hoax and yet they are masks (appropriately) on the same face seeking the same outcome. Those pushing the global warming myth through the Club of Rome and other Cult agencies are driving the lies about 'Covid' – Bill Gates is an obvious one, but they are endless. Why would the same people be involved in both when they are clearly not connected? Oh, but they

are. Common themes with personnel are matched by common goals. The 'solutions' to both 'problems' are centralisation of global power to impose the will of the few on the many to 'save' humanity from 'Covid' and save the planet from an 'existential threat' (we need 'zero Covid' and 'zero carbon emissions'). These, in turn, connect with the 'dot' of globalisation which was coined to describe the centralisation of global power in every area of life through incessant political and corporate expansion, trading blocks and superstates like the European Union. If you are the few and you want to control the many you have to centralise power and decision-making. The more you centralise power the more power the few at the centre will have over the many; and the more that power is centralised the more power those at the centre have to centralise even quicker. The momentum of centralisation gets faster and faster which is exactly the process we have witnessed. In this way the hoaxed 'pandemic' and the fakery of human-caused global warming serve the interests of globalisation and the seizure of global power in the hands of the Cult inner-circle which is behind 'Covid', 'climate change' *and* globalisation. At this point random 'dots' become a clear and obvious picture or pattern.

Klaus Schwab, the classic Bond villain who founded the Cult's Gates-funded World Economic Forum, published a book in 2020, *The Great Reset*, in which he used the 'problem' of 'Covid' to justify a total transformation of human society to 'save' humanity from 'climate change'. Schwab said: 'The pandemic represents a rare but narrow window of opportunity to reflect, reimagine, and reset our world.' What he didn't mention is that the Cult he serves is behind both hoaxes as I show in my book *The Answer*. He and the Cult don't have to reimagine the world. They know precisely what they want and that's why they destroyed human society with 'Covid' to 'build back better' in their grand design. Their job is not to imagine, but to get humanity to imagine and agree with their plans while believing it's all random. It must be pure coincidence that 'The Great Reset' has long been the Cult's code name for the global imposition of fascism and replaced previous code-names of the 'New World

Order' used by Cult frontmen like Father George Bush and the 'New Order of the Ages' which emerged from Freemasonry and much older secret societies. New Order of the Ages appears on the reverse of the Great Seal of the United States as 'Novus ordo seclorum' underneath the Cult symbol used since way back of the pyramid and all seeing-eye (Fig 3). The pyramid is the hierarchy of human control headed by the illuminated eye that symbolises the force behind the Cult which I will expose in later chapters. The term 'Annuit Coeptis' translates as 'He favours our undertaking'. We are told the 'He' is the Christian god, but 'He' is not as I will be explaining.

Figure 3: The all-seeing eye of the Cult 'god' on the Freemason-designed Great Seal of the United States and also on the dollar bill.

Having you on

Two major Cult techniques of perceptual manipulation that relate to all this are what I have called since the 1990s Problem-Reaction-Solution (PRS) and the Totalitarian Tiptoe (TT). They can be uncovered by the inquiring mind with a simple question: Who benefits? The answer usually identifies the perpetrators of a given action or happening through the concept of 'he who most benefits from a crime is the one most likely to have committed it'. The Latin 'Cue bono?' – Who benefits? – is widely attributed to the Roman orator and statesman Marcus Tullius Cicero. No wonder it goes back so far when the concept has been relevant to human behaviour since

history was recorded. Problem-Reaction-Solution is the technique used to manipulate us every day by covertly creating a problem (or the illusion of one) and offering the solution to the problem (or the illusion of one). In the first phase you create the problem and blame someone or something else for why it has happened. This may relate to a financial collapse, terrorist attack, war, global warming or pandemic, anything in fact that will allow you to impose the 'solution' to change society in the way you desire at that time. The 'problem' doesn't have to be real. PRS is manipulation of perception and all you need is the population to believe the problem is real. Human-caused global warming and the 'Covid pandemic' only have to be *perceived* to be real for the population to accept the 'solutions' of authority. I refer to this technique as NO-Problem-Reaction-Solution. Billions did not meekly accept house arrest from early 2020 because there was a real deadly 'Covid pandemic' but because they perceived – believed – that to be the case. The antidote to Problem-Reaction-Solution is to ask who benefits from the proposed solution. Invariably it will be anyone who wants to justify more control through deletion of freedom and centralisation of power and decision-making.

The two world wars were Problem-Reaction-Solutions that transformed and realigned global society. Both were manipulated into being by the Cult as I have detailed in books since the mid-1990s. They dramatically centralised global power, especially World War Two, which led to the United Nations and other global bodies thanks to the overt and covert manipulations of the Rockefeller family and other Cult bloodlines like the Rothschilds. The UN is a stalking horse for full-blown world government that I will come to shortly. The land on which the UN building stands in New York was donated by the Rockefellers and the same Cult family was behind Big Pharma scalpel and drug 'medicine' and the creation of the World Health Organization as part of the UN. They have been stalwarts of the eugenics movement and funded Hitler's race-purity expert' Ernst Rudin. The human-caused global warming hoax has been orchestrated by the Club of Rome through the UN which is

manufacturing both the 'problem' through its Intergovernmental Panel on Climate Change and imposing the 'solution' through its Agenda 21 and Agenda 2030 which demand the total centralisation of global power to 'save the world' from a climate hoax the United Nations is itself perpetrating. What a small world the Cult can be seen to be particularly among the inner circles. The bedfellow of Problem-Reaction-Solution is the Totalitarian Tiptoe which became the Totalitarian Sprint in 2020. The technique is fashioned to hide the carefully-coordinated behind the cover of apparently random events. You start the sequence at 'A' and you know you are heading for 'Z'. You don't want people to know that and each step on the journey is presented as a random happening while all the steps strung together lead in the same direction. The speed may have quickened dramatically in recent times, but you can still see the incremental approach of the Tiptoe in the case of 'Covid' as each new imposition takes us deeper into fascism. Tell people they have to do this or that to get back to 'normal', then this and this and this. With each new demand adding to the ones that went before the population's freedom is deleted until it disappears. The spider wraps its web around the flies more comprehensively with each new diktat. I'll highlight this in more detail when I get to the 'Covid' hoax and how it has been pulled off. Another prime example of the Totalitarian Tiptoe is how the Cult-created European Union went from a 'free-trade zone' to a centralised bureaucratic dictatorship through the Tiptoe of incremental centralisation of power until nations became mere administrative units for Cult-owned dark suits in Brussels.

The antidote to ignorance is knowledge which the Cult seeks vehemently to deny us, but despite the systematic censorship to that end the Renegade Mind can overcome this by vociferously seeking out the facts no matter the impediments put in the way. There is also a method of thinking and perceiving – *knowing* – that doesn't even need names, dates, place-type facts to identify the patterns that reveal the story. I'll get to that in the final chapter. All you need to know about the manipulation of human society and to what end is still out there – *at the time of writing* – in the form of books, videos

and websites for those that really want to breach the walls of programmed perception. To access this knowledge requires the abandonment of the mainstream media as a source of information in the awareness that this is owned and controlled by the Cult and therefore promotes mass perceptions that suit the Cult. Mainstream media lies all day, every day. That is its function and very reason for being. Where it does tell the truth, here and there, is only because the truth and the Cult agenda very occasionally coincide. If you look for fact and insight to the BBC, CNN and virtually all the rest of them you are asking to be conned and perceptually programmed.

Know the outcome and you'll see the journey

Events seem random when you have no idea where the world is being taken. Once you do the random becomes the carefully planned. Know the outcome and you'll see the journey is a phrase I have been using for a long time to give context to daily happenings that appear unconnected. Does a problem, or illusion of a problem, trigger a proposed 'solution' that further drives society in the direction of the outcome? Invariably the answer will be yes and the random – *abracadabra* – becomes the clearly coordinated. So what is this outcome that unlocks the door to a massively expanded understanding of daily events? I will summarise its major aspects – the fine detail is in my other books – and those new to this information will see that the world they thought they were living in is a very different place. The foundation of the Cult agenda is the incessant centralisation of power and all such centralisation is ultimately in pursuit of Cult control on a global level. I have described for a long time the planned world structure of top-down dictatorship as the Hunger Games Society. The term obviously comes from the movie series which portrayed a world in which a few living in military-protected hi-tech luxury were the overlords of a population condemned to abject poverty in isolated 'sectors' that were not allowed to interact. 'Covid' lockdowns and travel bans anyone? The 'Hunger Games' pyramid of structural control has the inner circle of the Cult at the top with pretty much the entire

population at the bottom under their control through dependency for survival on the Cult. The whole structure is planned to be protected and enforced by a military-police state (Fig 4).

Here you have the reason for the global lockdowns of the fake pandemic to coldly destroy independent incomes and livelihoods and make everyone dependent on the 'state' (the Cult that controls the 'states'). I have warned in my books for many years about the plan to introduce a 'guaranteed income' – a barely survivable pittance – designed to impose dependency when employment was destroyed by AI technology and now even more comprehensively at great speed by the 'Covid' scam. Once the pandemic was played and lockdown consequences began to delete independent income the authorities began to talk right on cue about the need for a guaranteed income and a 'Great Reset'. Guaranteed income will be presented as benevolent governments seeking to help a desperate people – desperate as a direct result of actions of the same governments. The truth is that such payments are a trap. You will only get them if you do exactly what the authorities demand including mass vaccination (genetic manipulation). We have seen this theme already in Australia where those dependent on government benefits have them reduced if parents don't agree to have their children vaccinated according to an insane health-destroying government-dictated schedule. Calculated economic collapse applies to governments as well as people. The Cult wants rid of countries through the creation of a world state with countries broken up into regions ruled by a world government and super states like the European Union. Countries must be bankrupted, too, to this end and it's being achieved by the trillions in 'rescue packages' and furlough payments, trillions in lost taxation, and money-no-object spending on 'Covid' including constant all-medium advertising (programming) which has made the media dependent on government for much of its income. The day of reckoning is coming – as planned – for government spending and given that it has been made possible by printing money and not by production/taxation there is inflation on the way that has the

potential to wipe out monetary value. In that case there will be no need for the Cult to steal your money. It just won't be worth anything (see the German Weimar Republic before the Nazis took over). Many have been okay with lockdowns while getting a percentage of their income from so-called furlough payments without having to work. Those payments are dependent, however, on people having at least a theoretical job with a business considered non-essential and ordered to close. As these business go under because they are closed by lockdown after lockdown the furlough stops and it will for everyone eventually. Then what? The 'then what?' is precisely the idea.

Figure 4: The Hunger Games Society structure I have long warned was planned and now the 'Covid' hoax has made it possible. This is the real reason for lockdowns.

Hired hands

Between the Hunger Games Cult elite and the dependent population is planned to be a vicious military-police state (a fusion of the two into one force). This has been in the making for a long time with police looking ever more like the military and carrying weapons to match. The pandemic scam has seen this process accelerate so fast as

lockdown house arrest is brutally enforced by carefully recruited fascist minds and gormless system-servers. The police and military are planned to merge into a centrally-directed world army in a global structure headed by a world government which wouldn't be elected even by the election fixes now in place. The world army is not planned even to be human and instead wars would be fought, primarily against the population, using robot technology controlled by artificial intelligence. I have been warning about this for decades and now militaries around the world are being transformed by this very AI technology. The global regime that I describe is a particular form of fascism known as a technocracy in which decisions are not made by clueless and co-opted politicians but by unelected technocrats – scientists, engineers, technologists and bureaucrats. Cult-owned-and-controlled Silicon Valley giants are examples of technocracy and they already have far more power to direct world events than governments. They are with their censorship *selecting* governments. I know that some are calling the 'Great Reset' a Marxist communist takeover, but fascism and Marxism are different labels for the same tyranny. Tell those who lived in fascist Germany and Stalinist Russia that there was a difference in the way their freedom was deleted and their lives controlled. I could call it a fascist technocracy or a Marxist technocracy and they would be equally accurate. The Hunger Games society with its world government structure would oversee a world army, world central bank and single world cashless currency imposing its will on a microchipped population (Fig 5). Scan its different elements and see how the illusory pandemic is forcing society in this very direction at great speed. Leaders of 23 countries and the World Health Organization (WHO) backed the idea in March, 2021, of a global treaty for 'international cooperation' in 'health emergencies' and nations should 'come together as a global community for peaceful cooperation that extends beyond this crisis'. Cut the Orwellian bullshit and this means another step towards global government. The plan includes a cashless digital money system that I first warned about in 1993. Right at the start of 'Covid' the deeply corrupt Tedros

Adhanom Ghebreyesus, the crooked and merely gofer 'head' of the World Health Organization, said it was possible to catch the 'virus' by touching cash and it was better to use cashless means. The claim was ridiculous nonsense and like the whole 'Covid' mind-trick it was nothing to do with 'health' and everything to do with pushing every aspect of the Cult agenda. As a result of the Tedros lie the use of cash has plummeted. The Cult script involves a single world digital currency that would eventually be technologically embedded in the body. China is a massive global centre for the Cult and if you watch what is happening there you will know what is planned for everywhere. The Chinese government is developing a digital currency which would allow fines to be deducted immediately via AI for anyone caught on camera breaking its fantastic list of laws and the money is going to be programmable with an expiry date to ensure that no one can accrue wealth except the Cult and its operatives.

Figure 5: The structure of global control the Cult has been working towards for so long and this has been enormously advanced by the 'Covid' illusion.

Serfdom is so smart

The Cult plan is far wider, extreme, and more comprehensive than even most conspiracy researchers appreciate and I will come to the true depths of deceit and control in the chapters 'Who controls the

Cult?' and 'Escaping Wetiko'. Even the world that we know is crazy enough. We are being deluged with ever more sophisticated and controlling technology under the heading of 'smart'. We have smart televisions, smart meters, smart cards, smart cars, smart driving, smart roads, smart pills, smart patches, smart watches, smart skin, smart borders, smart pavements, smart streets, smart cities, smart communities, smart environments, smart growth, smart planet ... smart *everything* around us. Smart technologies and methods of operation are designed to interlock to create a global Smart Grid connecting the entirety of human society including human minds to create a centrally-dictated 'hive' mind. 'Smart cities' is code for densely-occupied megacities of total surveillance and control through AI. Ever more destructive frequency communication systems like 5G have been rolled out without any official testing for health and psychological effects (colossal). 5G/6G/7G systems are needed to run the Smart Grid and each one becomes more destructive of body and mind. Deleting independent income is crucial to forcing people into these AI-policed prisons by ending private property ownership (except for the Cult elite). The Cult's Great Reset now openly foresees a global society in which no one will own any possessions and everything will be rented while the Cult would own literally everything under the guise of government and corporations. The aim has been to use the lockdowns to destroy sources of income on a mass scale and when the people are destitute and in unrepayable amounts of debt (problem) Cult assets come forward with the pledge to write-off debt in return for handing over all property and possessions (solution). Everything – literally everything including people – would be connected to the Internet via AI. I was warning years ago about the coming Internet of Things (IoT) in which all devices and technology from your car to your fridge would be plugged into the Internet and controlled by AI. Now we are already there with much more to come. The next stage is the Internet of Everything (IoE) which is planned to include the connection of AI to the human brain and body to replace the human mind with a centrally-controlled AI mind. Instead of perceptions

being manipulated through control of information and censorship those perceptions would come direct from the Cult through AI. What do you think? You think whatever AI decides that you think. In human terms there would be no individual 'think' any longer. Too incredible? The ravings of a lunatic? Not at all. Cult-owned crazies in Silicon Valley have been telling us the plan for years without explaining the real motivation and calculated implications. These include Google executive and 'futurist' Ray Kurzweil who highlights the year 2030 for when this would be underway. He said:

> Our thinking ... will be a hybrid of biological and non-biological thinking ... humans will be able to extend their limitations and 'think in the cloud' ... We're going to put gateways to the cloud in our brains ... We're going to gradually merge and enhance ourselves ... In my view, that's the nature of being human – we transcend our limitations.
>
> As the technology becomes vastly superior to what we are then the small proportion that is still human gets smaller and smaller and smaller until it's just utterly negligible.

The sales-pitch of Kurzweil and Cult-owned Silicon Valley is that this would make us 'super-human' when the real aim is to make us post-human and no longer 'human' in the sense that we have come to know. The entire global population would be connected to AI and become the centrally-controlled 'hive-mind' of externally-delivered perceptions. The Smart Grid being installed to impose the Cult's will on the world is being constructed to allow particular locations – even one location – to control the whole global system. From these prime control centres, which absolutely include China and Israel, anything connected to the Internet would be switched on or off and manipulated at will. Energy systems could be cut, communication via the Internet taken down, computer-controlled driverless autonomous vehicles driven off the road, medical devices switched off, the potential is limitless given how much AI and Internet connections now run human society. We have seen nothing yet if we allow this to continue. Autonomous vehicle makers are working with law enforcement to produce cars designed to automatically pull over if they detect a police or emergency vehicle flashing from up to 100 feet away. At a police stop the car would be unlocked and the

window rolled down automatically. Vehicles would only take you where the computer (the state) allowed. The end of petrol vehicles and speed limiters on all new cars in the UK and EU from 2022 are steps leading to electric computerised transport over which ultimately you have no control. The picture is far bigger even than the Cult global network or web and that will become clear when I get to the nature of the 'spider'. There is a connection between all these happenings and the instigation of DNA-manipulating 'vaccines' (which aren't 'vaccines') justified by the 'Covid' hoax. That connection is the unfolding plan to transform the human body from a biological to a synthetic biological state and this is why synthetic biology is such a fast-emerging discipline of mainstream science. 'Covid vaccines' are infusing self-replicating synthetic genetic material into the cells to cumulatively take us on the Totalitarian Tiptoe from Human 1.0 to the synthetic biological Human 2.0 which will be physically and perceptually attached to the Smart Grid to one hundred percent control every thought, perception and deed. Humanity needs to wake up and *fast*.

This is the barest explanation of where the 'outcome' is planned to go but it's enough to see the journey happening all around us. Those new to this information will already see 'Covid' in a whole new context. I will add much more detail as we go along, but for the minutiae evidence see my mega-works, *The Answer, The Trigger* and *Everything You Need to Know But Have Never Been Told*.

Now – how does a Renegade Mind see the 'world'?

CHAPTER TWO

Renegade Perception

It is one thing to be clever and another to be wise
George R.R. Martin

A simple definition of the difference between a programmed mind and a Renegade Mind would be that one sees only dots while the other connects them to see the picture. Reading reality with accuracy requires the observer to (a) know the planned outcome and (b) realise that everything, but *everything*, is connected.

The entirety of infinite reality is connected – that's its very nature – and with human society an expression of infinite reality the same must apply. Simple cause and effect is a connection. The effect is triggered by the cause and the effect then becomes the cause of another effect. Nothing happens in isolation because it *can't*. Life in whatever reality is simple choice and consequence. We make choices and these lead to consequences. If we don't like the consequences we can make different choices and get different consequences which lead to other choices and consequences. The choice and the consequence are not only connected they are indivisible. You can't have one without the other as an old song goes. A few cannot control the world unless those being controlled allow that to happen – cause and effect, choice and consequence. Control – who has it and who doesn't – is a two-way process, a symbiotic relationship, involving the controller and controlled. 'They took my freedom away!!' Well, yes, but you also gave it to them. Humanity is

subjected to mass control because humanity has acquiesced to that control. This is all cause and effect and literally a case of give and take. In the same way world events of every kind are connected and the Cult works incessantly to sell the illusion of the random and coincidental to maintain the essential (to them) perception of dots that hide the picture. Renegade Minds know this and constantly scan the world for patterns of connection. This is absolutely pivotal in understanding the happenings in the world and without that perspective clarity is impossible. First you know the planned outcome and then you identify the steps on the journey – the day-by-day apparently random which, when connected in relation to the outcome, no longer appear as individual events, but as the proverbial *chain* of events leading in the same direction. I'll give you some examples:

Political puppet show

We are told to believe that politics is 'adversarial' in that different parties with different beliefs engage in an endless tussle for power. There may have been some truth in that up to a point – and only a point – but today divisions between 'different' parties are rhetorical not ideological. Even the rhetorical is fusing into one-speak as the parties eject any remaining free thinkers while others succumb to the ever-gathering intimidation of anyone with the 'wrong' opinion. The Cult is not a new phenomenon and can be traced back thousands of years as my books have documented. Its intergenerational initiates have been manipulating events with increasing effect the more that global power has been centralised. In ancient times the Cult secured control through the system of monarchy in which 'special' bloodlines (of which more later) demanded the right to rule as kings and queens simply by birthright and by vanquishing others who claimed the same birthright. There came a time, however, when people had matured enough to see the unfairness of such tyranny and demanded a say in who governed them. Note the word – *governed* them. Not served them – *governed* them, hence government defined as 'the political direction and control exercised over the

actions of the members, citizens, or inhabitants of communities, societies, and states; direction of the affairs of a state, community, etc.' Governments exercise control over rather than serve just like the monarchies before them. Bizarrely there are still countries like the United Kingdom which are ruled by a monarch *and* a government that officially answers to the monarch. The UK head of state and that of Commonwealth countries such as Canada, Australia and New Zealand is 'selected' by who in a *single family* had unprotected sex with whom and in what order. Pinch me it can't be true. Ouch! Shit, it is. The demise of monarchies in most countries offered a potential vacuum in which some form of free and fair society could arise and the Cult had that base covered. Monarchies had served its interests but they couldn't continue in the face of such widespread opposition and, anyway, replacing a 'royal' dictatorship that people could see with a dictatorship 'of the people' hiding behind the concept of 'democracy' presented far greater manipulative possibilities and ways of hiding coordinated tyranny behind the illusion of 'freedom'.

Democracy is quite wrongly defined as government selected by the population. This is not the case at all. It is government selected by *some* of the population (and then only in theory). This 'some' doesn't even have to be the majority as we have seen so often in first-past-the-post elections in which the so-called majority party wins fewer votes than the 'losing' parties combined. Democracy can give total power to a party in government from a minority of the votes cast. It's a sleight of hand to sell tyranny as freedom. Seventy-four million Trump-supporting Americans didn't vote for the 'Democratic' Party of Joe Biden in the distinctly dodgy election in 2020 and yet far from acknowledging the wishes and feelings of that great percentage of American society the Cult-owned Biden government set out from day one to destroy them and their right to a voice and opinion. Empty shell Biden and his Cult handlers said they were doing this to 'protect democracy'. Such is the level of lunacy and sickness to which politics has descended. Connect the dots and relate them to the desired outcome – a world government run by self-appointed technocrats and no longer even elected

politicians. While operating through its political agents in government the Cult is at the same time encouraging public distain for politicians by putting idiots and incompetents in theoretical power on the road to deleting them. The idea is to instil a public reaction that says of the technocrats: 'Well, they couldn't do any worse than the pathetic politicians.' It's all about controlling perception and Renegade Minds can see through that while programmed minds cannot when they are ignorant of both the planned outcome and the manipulation techniques employed to secure that end. This knowledge can be learned, however, and fast if people choose to get informed.

Politics may at first sight appear very difficult to control from a central point. I mean look at the 'different' parties and how would you be able to oversee them all and their constituent parts? In truth, it's very straightforward because of their structure. We are back to the pyramid of imposition and acquiescence. Organisations are structured in the same way as the system as a whole. Political parties are not open forums of free expression. They are hierarchies. I was a national spokesman for the British Green Party which claimed to be a different kind of politics in which influence and power was devolved; but I can tell you from direct experience – and it's far worse now – that Green parties are run as hierarchies like all the others however much they may try to hide that fact or kid themselves that it's not true. A very few at the top of all political parties are directing policy and personnel. They decide if you are elevated in the party or serve as a government minister and to do that you have to be a yes man or woman. Look at all the maverick political thinkers who never ascended the greasy pole. If you want to progress within the party or reach 'high-office' you need to fall into line and conform. Exceptions to this are rare indeed. Should you want to run for parliament or Congress you have to persuade the local or state level of the party to select you and for that you need to play the game as dictated by the hierarchy. If you secure election and wish to progress within the greater structure you need to go on conforming to what is acceptable to those running the hierarchy

from the peak of the pyramid. Political parties are perceptual gulags and the very fact that there are party 'Whips' appointed to 'whip' politicians into voting the way the hierarchy demands exposes the ridiculous idea that politicians are elected to serve the people they are supposed to represent. Cult operatives and manipulation has long seized control of major parties that have any chance of forming a government and at least most of those that haven't. A new party forms and the Cult goes to work to infiltrate and direct. This has reached such a level today that you see video compilations of 'leaders' of all parties whether Democrats, Republicans, Conservative, Labour and Green parroting the same Cult mantra of 'Build Back Better' and the 'Great Reset' which are straight off the Cult song-sheet to describe the transformation of global society in response to the Cult-instigated hoaxes of the 'Covid pandemic' and human-caused 'climate change'. To see Caroline Lucas, the Green Party MP that I knew when I was in the party in the 1980s, speaking in support of plans proposed by Cult operative Klaus Schwab representing the billionaire global elite is a real head-shaker.

Many parties – one master

The party system is another mind-trick and was instigated to change the nature of the dictatorship by swapping 'royalty' for dark suits that people believed – though now ever less so – represented their interests. Understanding this trick is to realise that a single force (the Cult) controls all parties either directly in terms of the major ones or through manipulation of perception and ideology with others. You don't need to manipulate Green parties to demand your transformation of society in the name of 'climate change' when they are obsessed with the lie that this is essential to 'save the planet'. You just give them a platform and away they go serving your interests while believing they are being environmentally virtuous. America's political structure is a perfect blueprint for how the two or multi-party system is really a one-party state. The Republican Party is controlled from one step back in the shadows by a group made up of billionaires and their gofers known as neoconservatives or Neocons.

I have exposed them in fine detail in my books and they were the driving force behind the policies of the imbecilic presidency of Boy George Bush which included 9/11 (see *The Trigger* for a comprehensive demolition of the official story), the subsequent 'war on terror' (war *of* terror) and the invasions of Afghanistan and Iraq. The latter was a No-Problem-Reaction-Solution based on claims by Cult operatives, including Bush and British Prime Minister Tony Blair, about Saddam Hussein's 'weapons of mass destruction' which did not exist as war criminals Bush and Blair well knew.

Figure 6: Different front people, different parties – same control system.

The Democratic Party has its own 'Neocon' group controlling from the background which I call the 'Democons' and here's the penny-drop – the Neocons and Democons answer to the same masters one step further back into the shadows (Fig 6). At that level of the Cult the Republican and Democrat parties are controlled by the same people and no matter which is in power the Cult is in power. This is how it works in almost every country and certainly in Britain with Conservative, Labour, Liberal Democrat and Green parties now all on the same page whatever the rhetoric may be in their feeble attempts to appear different. Neocons operated at the time of Bush through a think tank called The Project for the New American Century which in September, 2000, published a document entitled *Rebuilding America's Defenses: Strategies, Forces, and Resources*

For a New Century demanding that America fight 'multiple, simultaneous major theatre wars' as a 'core mission' to force regime-change in countries including Iraq, Libya and Syria. Neocons arranged for Bush ('Republican') and Blair ('Labour Party') to front-up the invasion of Iraq and when they departed the Democons orchestrated the targeting of Libya and Syria through Barack Obama ('Democrat') and British Prime Minister David Cameron ('Conservative Party'). We have 'different' parties and 'different' people, but the same unfolding script. The more the Cult has seized the reigns of parties and personnel the more their policies have transparently pursued the same agenda to the point where the fascist 'Covid' impositions of the Conservative junta of Jackboot Johnson in Britain were opposed by the Labour Party because they were not fascist enough. The Labour Party is likened to the US Democrats while the Conservative Party is akin to a British version of the Republicans and on both sides of the Atlantic they all speak the same language and support the direction demanded by the Cult although some more enthusiastically than others. It's a similar story in country after country because it's all centrally controlled. Oh, but what about Trump? I'll come to him shortly. Political 'choice' in the 'party' system goes like this: You vote for Party A and they get into government. You don't like what they do so next time you vote for Party B and they get into government. You don't like what they do when it's pretty much the same as Party A and why wouldn't that be with both controlled by the same force? Given that only two, sometimes three, parties have any chance of forming a government to get rid of Party B that you don't like you have to vote again for Party A which … you don't like. This, ladies and gentlemen, is what they call 'democracy' which we are told – wrongly – is a term interchangeable with 'freedom'.

The cult of cults

At this point I need to introduce a major expression of the Global Cult known as Sabbatian-Frankism. Sabbatian is also spelt as Sabbatean. I will summarise here. I have published major exposés

and detailed background in other works. Sabbatian-Frankism combines the names of two frauds posing as 'Jewish' men, Sabbatai Zevi (1626-1676), a rabbi, black magician and occultist who proclaimed he was the Jewish messiah; and Jacob Frank (1726-1791), the Polish 'Jew', black magician and occultist who said he was the reincarnation of 'messiah' Zevi and biblical patriarch Jacob. They worked across two centuries to establish the Sabbatian-Frankist cult that plays a major, indeed central, role in the manipulation of human society by the Global Cult which has its origins much further back in history than Sabbatai Zevi. I should emphasise two points here in response to the shrill voices that will scream 'anti-Semitism': (1) Sabbatian-Frankists are NOT Jewish and only pose as such to hide their cult behind a Jewish façade; and (2) my information about this cult has come from Jewish sources who have long realised that their society and community has been infiltrated and taken over by interloper Sabbatian-Frankists. Infiltration has been the foundation technique of Sabbatian-Frankism from its official origin in the 17th century. Zevi's Sabbatian sect attracted a massive following described as the biggest messianic movement in Jewish history, spreading as far as Africa and Asia, and he promised a return for the Jews to the 'Promised Land' of Israel. Sabbatianism was not Judaism but an inversion of everything that mainstream Judaism stood for. So much so that this sinister cult would have a feast day when Judaism had a fast day and whatever was forbidden in Judaism the Sabbatians were encouraged and even commanded to do. This included incest and what would be today called Satanism. Members were forbidden to marry outside the sect and there was a system of keeping their children ignorant of what they were part of until they were old enough to be trusted not to unknowingly reveal anything to outsiders. The same system is employed to this day by the Global Cult in general which Sabbatian-Frankism has enormously influenced and now largely controls.

Zevi and his Sabbatians suffered a setback with the intervention by the Sultan of the Islamic Ottoman Empire in the Middle East and what is now the Republic of Turkey where Zevi was located. The

Sultan gave him the choice of proving his 'divinity', converting to Islam or facing torture and death. Funnily enough Zevi chose to convert or at least appear to. Some of his supporters were disillusioned and drifted away, but many did not with 300 families also converting – only in theory – to Islam. They continued behind this Islamic smokescreen to follow the goals, rules and rituals of Sabbatianism and became known as 'crypto-Jews' or the 'Dönmeh' which means 'to turn'. This is rather ironic because they didn't 'turn' and instead hid behind a fake Islamic persona. The process of appearing to be one thing while being very much another would become the calling card of Sabbatianism especially after Zevi's death and the arrival of the Satanist Jacob Frank in the 18th century when the cult became Sabbatian-Frankism and plumbed still new depths of depravity and infiltration which included – still includes – human sacrifice and sex with children. Wherever Sabbatians go paedophilia and Satanism follow and is it really a surprise that Hollywood is so infested with child abuse and Satanism when it was established by Sabbatian-Frankists and is still controlled by them? Hollywood has been one of the prime vehicles for global perceptual programming and manipulation. How many believe the version of 'history' portrayed in movies when it is a travesty and inversion (again) of the truth? Rabbi Marvin Antelman describes Frankism in his book, *To Eliminate the Opiate*, as 'a movement of complete evil' while Jewish professor Gershom Scholem said of Frank in *The Messianic Idea in Judaism*: 'In all his actions [he was] a truly corrupt and degenerate individual ... one of the most frightening phenomena in the whole of Jewish history.' Frank was excommunicated by traditional rabbis, as was Zevi, but Frank was undeterred and enjoyed vital support from the House of Rothschild, the infamous banking dynasty whose inner-core are Sabbatian-Frankists and not Jews. Infiltration of the Roman Church and Vatican was instigated by Frank with many Dönmeh 'turning' again to convert to Roman Catholicism with a view to hijacking the reins of power. This was the ever-repeating modus operandi and continues to be so. Pose as an advocate of the religion, culture or country that you want to control and then

manipulate your people into the positions of authority and influence largely as advisers, administrators and Svengalis for those that appear to be in power. They did this with Judaism, Christianity (Christian Zionism is part of this), Islam and other religions and nations until Sabbatian-Frankism spanned the world as it does today.

Sabbatian Saudis and the terror network

One expression of the Sabbatian-Frankist Dönmeh within Islam is the ruling family of Saudi Arabia, the House of Saud, through which came the vile distortion of Islam known as Wahhabism. This is the violent creed followed by terrorist groups like Al-Qaeda and ISIS or Islamic State. Wahhabism is the hand-chopping, head-chopping 'religion' of Saudi Arabia which is used to keep the people in a constant state of fear so the interloper House of Saud can continue to rule. Al-Qaeda and Islamic State were lavishly funded by the House of Saud while being created and directed by the Sabbatian-Frankist network in the United States that operates through the Pentagon, CIA and the government in general of whichever 'party'. The front man for the establishment of Wahhabism in the middle of the 18th century was a Sabbatian-Frankist 'crypto-Jew' posing as Islamic called Muhammad ibn Abd al-Wahhab. His daughter would marry the son of Muhammad bin Saud who established the first Saudi state before his death in 1765 with support from the British Empire. Bin Saud's successors would establish modern Saudi Arabia in league with the British and Americans in 1932 which allowed them to seize control of Islam's major shrines in Mecca and Medina. They have dictated the direction of Sunni Islam ever since while Iran is the major centre of the Shiite version and here we have the source of at least the public conflict between them. The Sabbatian network has used its Wahhabi extremists to carry out Problem-Reaction-Solution terrorist attacks in the name of 'Al-Qaeda' and 'Islamic State' to justify a devastating 'war on terror', ever-increasing surveillance of the population and to terrify people into compliance. Another insight of the Renegade Mind is the streetwise understanding that

just because a country, location or people are attacked doesn't mean that those apparently representing that country, location or people are not behind the attackers. Often they are *orchestrating* the attacks because of the societal changes that can be then justified in the name of 'saving the population from terrorists'.

I show in great detail in *The Trigger* how Sabbatian-Frankists were the real perpetrators of 9/11 and not '19 Arab hijackers' who were blamed for what happened. Observe what was justified in the name of 9/11 alone in terms of Middle East invasions, mass surveillance and control that fulfilled the demands of the Project for the New American Century document published by the Sabbatian Neocons. What appear to be enemies are on the deep inside players on the same Sabbatian team. Israel and Arab 'royal' dictatorships are all ruled by Sabbatians and the recent peace agreements between Israel and Saudi Arabia, the United Arab Emirates (UAE) and others are only making formal what has always been the case behind the scenes. Palestinians who have been subjected to grotesque tyranny since Israel was bombed and terrorised into existence in 1948 have never stood a chance. Sabbatian-Frankists have controlled Israel (so the constant theme of violence and war which Sabbatians love) and they have controlled the Arab countries that Palestinians have looked to for real support that never comes. 'Royal families' of the Arab world in Saudi Arabia, Bahrain, UAE, etc., are all Sabbatians with allegiance to the aims of the cult and not what is best for their Arabic populations. They have stolen the oil and financial resources from their people by false claims to be 'royal dynasties' with a genetic right to rule and by employing vicious militaries to impose their will.

Satanic 'illumination'

The Satanist Jacob Frank formed an alliance in 1773 with two other Sabbatians, Mayer Amschel Rothschild (1744-1812), founder of the Rothschild banking dynasty, and Jesuit-educated fraudulent Jew, Adam Weishaupt, and this led to the formation of the Bavarian Illuminati, firstly under another name, in 1776. The Illuminati would

be the manipulating force behind the French Revolution (1789-1799) and was also involved in the American Revolution (1775-1783) before and after the Illuminati's official creation. Weishaupt would later become (in public) a Protestant Christian in archetypal Sabbatian style. I read that his name can be decoded as Adam-Weishaupt or 'the first man to lead those who know'. He wasn't a leader in the sense that he was a subordinate, but he did lead those below him in a crusade of transforming human society that still continues today. The theme was confirmed as early as 1785 when a horseman courier called Lanz was reported to be struck by lighting and extensive Illuminati documents were found in his saddlebags. They made the link to Weishaupt and detailed the plan for world takeover. Current events with 'Covid' fascism have been in the making for a very long time. Jacob Frank was jailed for 13 years by the Catholic Inquisition after his arrest in 1760 and on his release he headed for Frankfurt, Germany, home city and headquarters of the House of Rothschild where the alliance was struck with Mayer Amschel Rothschild and Weishaupt. Rothschild arranged for Frank to be given the title of Baron and he became a wealthy nobleman with a big following of Jews in Germany, the Austro-Hungarian Empire and other European countries. Most of them would have believed he was on their side.

The name 'Illuminati' came from the Zohar which is a body of works in the Jewish mystical 'bible' called the Kabbalah. 'Zohar' is the foundation of Sabbatian-Frankist belief and in Hebrew 'Zohar' means 'splendour', 'radiance', 'illuminated', and so we have 'Illuminati'. They claim to be the 'Illuminated Ones' from their knowledge systematically hidden from the human population and passed on through generations of carefully-chosen initiates in the global secret society network or Cult. Hidden knowledge includes an awareness of the Cult agenda for the world and the nature of our collective reality that I will explore later. Cult 'illumination' is symbolised by the torch held by the Statue of Liberty which was gifted to New York by French Freemasons in Paris who knew exactly what it represents. 'Liberty' symbolises the goddess worshipped in

Babylon as Queen Semiramis or Ishtar. The significance of this will become clear. Notice again the ubiquitous theme of inversion with the Statue of 'Liberty' really symbolising mass control (Fig 7). A mirror-image statute stands on an island in the River Seine in Paris from where New York Liberty originated (Fig 8). A large replica of the Liberty flame stands on top of the Pont de l'Alma tunnel in Paris where Princess Diana died in a Cult ritual described in *The Biggest Secret*. Lucifer 'the light bringer' is related to all this (and much more as we'll see) and 'Lucifer' is a central figure in Sabbatian-Frankism and its associated Satanism. Sabbatians reject the Jewish Torah, or Pentateuch, the 'five books of Moses' in the Old Testament known as Genesis, Exodus, Leviticus, Numbers, and Deuteronomy which are claimed by Judaism and Christianity to have been dictated by 'God' to Moses on Mount Sinai. Sabbatians say these do not apply to them and they seek to replace them with the Zohar to absorb Judaism and its followers into their inversion which is an expression of a much greater global inversion. They want to delete all religions and force humanity to worship a one-world religion – Sabbatian Satanism that also includes worship of the Earth goddess. Satanic themes are being more and more introduced into mainstream society and while Christianity is currently the foremost target for destruction the others are planned to follow.

Figure 7: The Cult goddess of Babylon disguised as the Statue of Liberty holding the flame of Lucifer the 'light bringer'.

Figure 8: Liberty's mirror image in Paris where the New York version originated.

Marx brothers

Rabbi Marvin Antelman connects the Illuminati to the Jacobins in *To Eliminate the Opiate* and Jacobins were the force behind the French Revolution. He links both to the Bund der Gerechten, or League of the Just, which was the network that inflicted communism/Marxism on the world. Antelman wrote:

> The original inner circle of the Bund der Gerechten consisted of born Catholics, Protestants and Jews [Sabbatian-Frankist infiltrators], and those representatives of respective subdivisions formulated schemes for the ultimate destruction of their faiths. The heretical Catholics laid plans which they felt would take a century or more for the ultimate destruction of the church; the apostate Jews for the ultimate destruction of the Jewish religion.

Sabbatian-created communism connects into this anti-religion agenda in that communism does not allow for the free practice of religion. The Sabbatian 'Bund' became the International Communist Party and Communist League and in 1848 'Marxism' was born with the Communist Manifesto of Sabbatian assets Karl Marx and Friedrich Engels. It is absolutely no coincidence that Marxism, just a different name for fascist and other centrally-controlled tyrannies, is being imposed worldwide as a result of the 'Covid' hoax and nor that Marxist/fascist China was the place where the hoax originated. The reason for this will become very clear in the chapter 'Covid: The calculated catastrophe'. The so-called 'Woke' mentality has hijacked

traditional beliefs of the political left and replaced them with far-right make-believe 'social justice' better known as Marxism. Woke will, however, be swallowed by its own perceived 'revolution' which is really the work of billionaires and billionaire corporations feigning being 'Woke'. Marxism is being touted by Wokers as a replacement for 'capitalism' when we don't have 'capitalism'. We have cartelism in which the market is stitched up by the very Cult billionaires and corporations bankrolling Woke. Billionaires love Marxism which keeps the people in servitude while they control from the top. Terminally naïve Wokers think they are 'changing the world' when it's the Cult that is doing the changing and when they have played their vital part and become surplus to requirements they, too, will be targeted. The Illuminati-Jacobins were behind the period known as 'The Terror' in the French Revolution in 1793 and 1794 when Jacobin Maximillian de Robespierre and his Orwellian 'Committee of Public Safety' killed 17,000 'enemies of the Revolution' who had once been 'friends of the Revolution'. Karl Marx (1818-1883), whose Sabbatian creed of Marxism has cost the lives of at least 100 million people, is a hero once again to Wokers who have been systematically kept ignorant of real history by their 'education' programming. As a result they now promote a Sabbatian 'Marxist' abomination destined at some point to consume them. Rabbi Antelman, who spent decades researching the Sabbatian plot, said of the League of the Just and Karl Marx:

> Contrary to popular opinion Karl Marx did not originate the Communist Manifesto. He was paid for his services by the League of the Just, which was known in its country of origin, Germany, as the Bund der Geaechteten.

Antelman said the text attributed to Marx was the work of other people and Marx 'was only repeating what others already said'. Marx was 'a hired hack – lackey of the wealthy Illuminists'. Marx famously said that religion was the 'opium of the people' (part of the Sabbatian plan to demonise religion) and Antelman called his books, *To Eliminate the Opiate*. Marx was born Jewish, but his family converted to Christianity (Sabbatian modus operandi) and he

attacked Jews, not least in his book, *A World Without Jews*. In doing so he supported the Sabbatian plan to destroy traditional Jewishness and Judaism which we are clearly seeing today with the vindictive targeting of orthodox Jews by the Sabbatian government of Israel over 'Covid' laws. I don't follow any religion and it has done much damage to the world over centuries and acted as a perceptual straightjacket. Renegade Minds, however, are always asking *why* something is being done. It doesn't matter if they agree or disagree with what is happening – *why* is it happening is the question. The 'why?' can be answered with regard to religion in that religions create interacting communities of believers when the Cult wants to dismantle all discourse, unity and interaction (see 'Covid' lockdowns) and the ultimate goal is to delete all religions for a one-world religion of Cult Satanism worshipping their 'god' of which more later. We see the same 'why?' with gun control in America. I don't have guns and don't want them, but why is the Cult seeking to disarm the population at the same time that law enforcement agencies are armed to their molars and why has every tyrant in history sought to disarm people before launching the final takeover? They include Hitler, Stalin, Pol Pot and Mao who followed confiscation with violent seizing of power. You know it's a Cult agenda by the people who immediately race to the microphones to exploit dead people in multiple shootings. Ultra-Zionist Cult lackey Senator Chuck Schumer was straight on the case after ten people were killed in Boulder, Colorado in March, 2121. Simple rule … if Schumer wants it the Cult wants it and the same with his ultra-Zionist mate the wild-eyed Senator Adam Schiff. At the same time they were calling for the disarmament of Americans, many of whom live a long way from a police response, Schumer, Schiff and the rest of these pampered clowns were sitting on Capitol Hill behind a razor-wired security fence protected by thousands of armed troops in addition to their own armed bodyguards. Mom and pop in an isolated home? They're just potential mass shooters.

Zion Mainframe

Sabbatian-Frankists and most importantly the Rothschilds were behind the creation of 'Zionism', a political movement that demanded a Jewish homeland in Israel as promised by Sabbatai Zevi. The very symbol of Israel comes from the German meaning of the name Rothschild. Dynasty founder Mayer Amschel Rothschild changed the family name from Bauer to Rothschild, or 'Red-Shield' in German, in deference to the six-pointed 'Star of David' hexagram displayed on the family's home in Frankfurt. The symbol later appeared on the flag of Israel after the Rothschilds were centrally involved in its creation. Hexagrams are not a uniquely Jewish symbol and are widely used in occult ('hidden') networks often as a symbol for Saturn (see my other books for why). Neither are Zionism and Jewishness interchangeable. Zionism is a political movement and philosophy and not a 'race' or a people. Many Jews oppose Zionism and many non-Jews, including US President Joe Biden, call themselves Zionists as does Israel-centric Donald Trump. America's support for the Israel government is pretty much a gimme with ultra-Zionist billionaires and corporations providing fantastic and dominant funding for both political parties. Former Congresswoman Cynthia McKinney has told how she was approached immediately she ran for office to 'sign the pledge' to Israel and confirm that she would always vote in that country's best interests. All American politicians are approached in this way. Anyone who refuses will get no support or funding from the enormous and all-powerful Zionist lobby that includes organisations like mega-lobby group AIPAC, the American Israel Public Affairs Committee. Trump's biggest funder was ultra-Zionist casino and media billionaire Sheldon Adelson while major funders of the Democratic Party include ultra-Zionist George Soros and ultra-Zionist financial and media mogul, Haim Saban. Some may reel back at the suggestion that Soros is an Israel-firster (Sabbatian-controlled Israel-firster), but Renegade Minds watch the actions not the words and everywhere Soros donates his billions the Sabbatian agenda benefits. In the spirit of Sabbatian inversion Soros pledged $1 billion for a new university network to promote 'liberal values and tackle intolerance'. He made the announcement during his annual speech

at the Cult-owned World Economic Forum in Davos, Switzerland, in January, 2020, after his 'harsh criticism' of 'authoritarian rulers' around the world. You can only laugh at such brazen mendacity. How *he* doesn't laugh is the mystery. Translated from the Orwellian 'liberal values and tackle intolerance' means teaching non-white people to hate white people and for white people to loathe themselves for being born white. The reason for that will become clear.

The 'Anti-Semitism' fraud

Zionists support the Jewish homeland in the land of Palestine which has been the Sabbatian-Rothschild goal for so long, but not for the benefit of Jews. Sabbatians and their global Anti-Semitism Industry have skewed public and political opinion to equate opposing the violent extremes of Zionism to be a blanket attack and condemnation of all Jewish people. Sabbatians and their global Anti-Semitism Industry have skewed public and political opinion to equate opposing the violent extremes of Zionism to be a blanket attack and condemnation of all Jewish people. This is nothing more than a Sabbatian protection racket to stop legitimate investigation and exposure of their agendas and activities. The official definition of 'anti-Semitism' has more recently been expanded to include criticism of Zionism – a *political movement* – and this was done to further stop exposure of Sabbatian infiltrators who created Zionism as we know it today in the 19th century. Renegade Minds will talk about these subjects when they know the shit that will come their way. People must decide if they want to know the truth or just cower in the corner in fear of what others will say. Sabbatians have been trying to label me as 'anti-Semitic' since the 1990s as I have uncovered more and more about their background and agendas. Useless, gutless, fraudulent 'journalists' then just repeat the smears without question and on the day I was writing this section a pair of unquestioning repeaters called Ben Quinn and Archie Bland (how appropriate) outright called me an 'anti-Semite' in the establishment propaganda sheet, the London *Guardian*, with no supporting evidence. The

Sabbatian Anti-Semitism Industry said so and who are they to question that? They wouldn't dare. Ironically 'Semitic' refers to a group of languages in the Middle East that are almost entirely Arabic. 'Anti-Semitism' becomes 'anti-Arab' which if the consequences of this misunderstanding were not so grave would be hilarious. Don't bother telling Quinn and Bland. I don't want to confuse them, bless 'em. One reason I am dubbed 'anti-Semitic' is that I wrote in the 1990s that Jewish operatives (Sabbatians) were heavily involved in the Russian Revolution when Sabbatians overthrew the Romanov dynasty. This apparently made me 'anti-Semitic'. Oh, really? Here is a section from *The Trigger*:

British journalist Robert Wilton confirmed these themes in his 1920 book *The Last Days of the Romanovs* when he studied official documents from the Russian government to identify the members of the Bolshevik ruling elite between 1917 and 1919. The Central Committee included 41 Jews among 62 members; the Council of the People's Commissars had 17 Jews out of 22 members; and 458 of the 556 most important Bolshevik positions between 1918 and 1919 were occupied by Jewish people. Only 17 were Russian. Then there were the 23 Jews among the 36 members of the vicious Cheka Soviet secret police established in 1917 who would soon appear all across the country.

Professor Robert Service of Oxford University, an expert on 20th century Russian history, found evidence that ['Jewish'] Leon Trotsky had sought to make sure that Jews were enrolled in the Red Army and were disproportionately represented in the Soviet civil bureaucracy that included the Cheka which performed mass arrests, imprisonment and executions of 'enemies of the people'. A US State Department Decimal File (861.00/5339) dated November 13th, 1918, names [Rothschild banking agent in America] Jacob Schiff and a list of ultra-Zionists as funders of the Russian Revolution leading to claims of a 'Jewish plot', but the key point missed by all is they were not 'Jews' – they were Sabbatian-Frankists.

Britain's Winston Churchill made the same error by mistake or otherwise. He wrote in a 1920 edition of the *Illustrated Sunday Herald* that those behind the Russian revolution were part of a 'worldwide conspiracy for the overthrow of civilisation and for the reconstitution of society on the basis of arrested development, of envious malevolence, and impossible equality' (see 'Woke' today because that has been created by the same network). Churchill said there was no need to exaggerate the part played in the creation of Bolshevism and in the actual bringing about of the Russian

Revolution 'by these international and for the most part atheistical Jews' ['atheistical Jews' = Sabbatians]. Churchill said it is certainly a very great one and probably outweighs all others: 'With the notable exception of Lenin, the majority of the leading figures are Jews.' He went on to describe, knowingly or not, the Sabbatian modus operandi of placing puppet leaders nominally in power while they control from the background:

> Moreover, the principal inspiration and driving power comes from the Jewish leaders. Thus Tchitcherin, a pure Russian, is eclipsed by his nominal subordinate, Litvinoff, and the influence of Russians like Bukharin or Lunacharski cannot be compared with the power of Trotsky, or of Zinovieff, the Dictator of the Red Citadel (Petrograd), or of Krassin or Radek – all Jews. In the Soviet institutions the predominance of Jews is even more astonishing. And the prominent, if not indeed the principal, part in the system of terrorism applied by the Extraordinary Commissions for Combatting Counter-Revolution has been taken by Jews, and in some notable cases by Jewesses.

What I said about seriously disproportionate involvement in the Russian Revolution by Jewish 'revolutionaries' (Sabbatians) is provable fact, but truth is no defence against the Sabbatian Anti-Semitism Industry, its repeater parrots like Quinn and Bland, and the now breathtaking network of so-called 'Woke' 'anti-hate' groups with interlocking leaderships and funding which have the role of discrediting and silencing anyone who gets too close to exposing the Sabbatians. We have seen 'truth is no defence' confirmed in legal judgements with the Saskatchewan Human Rights Commission in Canada decreeing this: 'Truthful statements can be presented in a manner that would meet the definition of hate speech, and not all truthful statements must be free from restriction.' Most 'anti-hate' activists, who are themselves consumed by hatred, are too stupid and ignorant of the world to know how they are being used. They are far too far up their own virtue-signalling arses and it's far too dark for them to see anything.

The 'revolution' game

The background and methods of the 'Russian' Revolution are straight from the Sabbatian playbook seen in the French Revolution

and endless others around the world that appear to start as a revolution of the people against tyrannical rule and end up with a regime change to more tyrannical rule overtly or covertly. Wars, terror attacks and regime overthrows follow the Sabbatian cult through history with its agents creating them as Problem-Reaction-Solutions to remove opposition on the road to world domination. Sabbatian dots connect the Rothschilds with the Illuminati, Jacobins of the French Revolution, the 'Bund' or League of the Just, the International Communist Party, Communist League and the Communist Manifesto of Karl Marx and Friedrich Engels that would lead to the Rothschild-funded Russian Revolution. The sequence comes under the heading of 'creative destruction' when you advance to your global goal by continually destroying the status quo to install a new status quo which you then also destroy. The two world wars come to mind. With each new status quo you move closer to your planned outcome. Wars and mass murder are to Sabbatians a collective blood sacrifice ritual. They are obsessed with death for many reasons and one is that death is an inversion of life. Satanists and Sabbatians are obsessed with death and often target churches and churchyards for their rituals. Inversion-obsessed Sabbatians explain the use of inverted symbolism including the *inverted* pentagram and *inverted* cross. The inversion of the cross has been related to targeting Christianity, but the cross was a religious symbol long before Christianity and its inversion is a statement about the Sabbatian mentality and goals more than any single religion.

Sabbatians operating in Germany were behind the rise of the occult-obsessed Nazis and the subsequent Jewish exodus from Germany and Europe to Palestine and the United States after World War Two. The Rothschild dynasty was at the forefront of this both as political manipulators and by funding the operation. Why would Sabbatians help to orchestrate the horrors inflicted on Jews by the Nazis and by Stalin after they organised the Russian Revolution? Sabbatians hate Jews and their religion, that's why. They pose as Jews and secure positions of control within Jewish society and play the 'anti-Semitism' card to protect themselves from exposure

through a global network of organisations answering to the Sabbatian-created-and-controlled globe-spanning intelligence network that involves a stunning web of military-intelligence operatives and operations for a tiny country of just nine million. Among them are Jewish assets who are not Sabbatians but have been convinced by them that what they are doing is for the good of Israel and the Jewish community to protect them from what they have been programmed since childhood to believe is a Jew-hating hostile world. The Jewish community is just a highly convenient cover to hide the true nature of Sabbatians. Anyone getting close to exposing their game is accused by Sabbatian place-people and gofers of 'anti-Semitism' and claiming that all Jews are part of a plot to take over the world. I am not saying that. I am saying that Sabbatians – the *real* Jew-haters – have infiltrated the Jewish community to use them both as a cover and an 'anti-Semitic' defence against exposure. Thus we have the Anti-Semitism Industry targeted researchers in this way and most Jewish people think this is justified and genuine. They don't know that their 'Jewish' leaders and institutions of state, intelligence and military are not controlled by Jews at all, but cultists and stooges of Sabbatian-Frankism. I once added my name to a pro-Jewish freedom petition online and the next time I looked my name was gone and text had been added to the petition blurb to attack me as an 'anti-Semite' such is the scale of perceptual programming.

Moving on America

I tell the story in *The Trigger* and a chapter called 'Atlantic Crossing' how particularly after Israel was established the Sabbatians moved in on the United States and eventually grasped control of government administration, the political system via both Democrats and Republicans, the intelligence community like the CIA and National Security Agency (NSA), the Pentagon and mass media. Through this seriously compartmentalised network Sabbatians and their operatives in Mossad, Israeli Defense Forces (IDF) and US agencies pulled off 9/11 and blamed it on 19 'Al-Qaeda hijackers' dominated by men from, or connected to, Sabbatian-ruled Saudi

Arabia. The '19' were not even on the planes let alone flew those big passenger jets into buildings while being largely incompetent at piloting one-engine light aircraft. 'Hijacker' Hani Hanjour who is said to have flown American Airlines Flight 77 into the Pentagon with a turn and manoeuvre most professional pilots said they would have struggled to do was banned from renting a small plane by instructors at the Freeway Airport in Bowie, Maryland, just *six weeks* earlier on the grounds that he was an incompetent pilot. The Jewish population of the world is just 0.2 percent with even that almost entirely concentrated in Israel (75 percent Jewish) and the United States (around two percent). This two percent and globally 0.2 percent refers to *Jewish* people and not Sabbatian interlopers who are a fraction of that fraction. What a sobering thought when you think of the fantastic influence on world affairs of tiny Israel and that the Project for the New America Century (PNAC) which laid out the blueprint in September, 2000, for America's war on terror and regime change wars in Iraq, Libya and Syria was founded and dominated by Sabbatians known as 'Neocons'. The document conceded that this plan would not be supported politically or publicly without a major attack on American soil and a Problem-Reaction-Solution excuse to send troops to war across the Middle East. Sabbatian Neocons said:

> ... [The] process of transformation ... [war and regime change] ... is likely to be a long one, absent some catastrophic and catalysing event – like a new Pearl Harbor.

Four months later many of those who produced that document came to power with their inane puppet George Bush from the long-time Sabbatian Bush family. They included Sabbatian Dick Cheney who was officially vice-president, but really de-facto president for the entirety of the 'Bush' government. Nine months after the 'Bush' inauguration came what Bush called at the time 'the Pearl Harbor of the 21st century' and with typical Sabbatian timing and symbolism 2001 was the 60th anniversary of the attack in 1941 by the Japanese Air Force on Pearl Harbor, Hawaii, which allowed President Franklin Delano Roosevelt to take the United States into a Sabbatian-

instigated Second World War that he said in his election campaign that he never would. The evidence is overwhelming that Roosevelt and his military and intelligence networks knew the attack was coming and did nothing to stop it, but they did make sure that America's most essential naval ships were not in Hawaii at the time. Three thousand Americans died in the Pearl Harbor attacks as they did on September 11th. By the 9/11 year of 2001 Sabbatians had widely infiltrated the US government, military and intelligence operations and used their compartmentalised assets to pull off the 'Al-Qaeda' attacks. If you read *The Trigger* it will blow your mind to see the utterly staggering concentration of 'Jewish' operatives (Sabbatian infiltrators) in essential positions of political, security, legal, law enforcement, financial and business power before, during, and after the attacks to make them happen, carry them out, and then cover their tracks – and I do mean *staggering* when you think of that 0.2 percent of the world population and two percent of Americans which are Jewish while Sabbatian infiltrators are a fraction of that. A central foundation of the 9/11 conspiracy was the hijacking of government, military, Air Force and intelligence computer systems in real time through 'back-door' access made possible by Israeli (Sabbatian) 'cyber security' software. Sabbatian-controlled Israel is on the way to rivalling Silicon Valley for domination of cyberspace and is becoming the dominant force in cyber-security which gives them access to entire computer systems and their passcodes across the world. Then add to this that Zionists head (officially) Silicon Valley giants like Google (Larry Page and Sergey Brin), Google-owned YouTube (Susan Wojcicki), Facebook (Mark Zuckerberg and Sheryl Sandberg), and Apple (Chairman Arthur D. Levinson), and that ultra-Zionist hedge fund billionaire Paul Singer has a $1 billion stake in Twitter which is only nominally headed by 'CEO' pothead Jack Dorsey. As cable news host Tucker Carlson said of Dorsey: 'There used to be debate in the medical community whether dropping a ton of acid had permanent effects and I think that debate has now ended.' Carlson made the comment after Dorsey told a hearing on Capitol Hill (if you cut through his bullshit) that he

believed in free speech so long as he got to decide what you can hear and see. These 'big names' of Silicon Valley are only front men and women for the Global Cult, not least the Sabbatians, who are the true controllers of these corporations. Does anyone still wonder why these same people and companies have been ferociously censoring and banning people (like me) for exposing any aspect of the Cult agenda and especially the truth about the 'Covid' hoax which Sabbatians have orchestrated?

The Jeffrey Epstein paedophile ring was a Sabbatian operation. He was officially 'Jewish' but he was a Sabbatian and women abused by the ring have told me about the high number of 'Jewish' people involved. The Epstein horror has Sabbatian written all over it and matches perfectly their modus operandi and obsession with sex and ritual. Epstein was running a Sabbatian blackmail ring in which famous people with political and other influence were provided with young girls for sex while everything was being filmed and recorded on hidden cameras and microphones at his New York house, Caribbean island and other properties. Epstein survivors have described this surveillance system to me and some have gone public. Once the famous politician or other figure knew he or she was on video they tended to do whatever they were told. Here we go again …when you've got them by the balls their hearts and minds will follow. Sabbatians use this blackmail technique on a wide scale across the world to entrap politicians and others they need to act as demanded. Epstein's private plane, the infamous 'Lolita Express', had many well-known passengers including Bill Clinton while Bill Gates has flown on an Epstein plane and met with him four years after Epstein had been jailed for paedophilia. They subsequently met many times at Epstein's home in New York according to a witness who was there. Epstein's infamous side-kick was Ghislaine Maxwell, daughter of Mossad agent and ultra-Zionist mega-crooked British businessman, Bob Maxwell, who at one time owned the *Daily Mirror* newspaper. Maxwell was murdered at sea on his boat in 1991 by Sabbatian-controlled Mossad when he became a liability with his

business empire collapsing as a former Mossad operative has confirmed (see *The Trigger*).

Money, money, money, funny money ...

Before I come to the Sabbatian connection with the last three US presidents I will lay out the crucial importance to Sabbatians of controlling banking and finance. Sabbatian Mayer Amschel Rothschild set out to dominate this arena in his family's quest for total global control. What is freedom? It is, in effect, choice. The more choices you have the freer you are and the fewer your choices the more you are enslaved. In the global structure created over centuries by Sabbatians the biggest decider and restrictor of choice is … money. Across the world if you ask people what they would like to do with their lives and why they are not doing that they will reply 'I don't have the money'. This is the idea. A global elite of multi-billionaires are described as 'greedy' and that is true on one level; but control of money – who has it and who doesn't – is not primarily about greed. It's about control. Sabbatians have seized ever more control of finance and sucked the wealth of the world out of the hands of the population. We talk now, after all, about the 'One-percent' and even then the wealthiest are a lot fewer even than that. This has been made possible by a money scam so outrageous and so vast it could rightly be called the scam of scams founded on creating 'money' out of nothing and 'loaning' that with interest to the population. Money out of nothing is called 'credit'. Sabbatians have asserted control over governments and banking ever more completely through the centuries and secured financial laws that allow banks to lend hugely more than they have on deposit in a confidence trick known as fractional reserve lending. Imagine if you could lend money that doesn't exist and charge the recipient interest for doing so. You would end up in jail. Bankers by contrast end up in mansions, private jets, Malibu and Monaco.

Banks are only required to keep a fraction of their deposits and wealth in their vaults and they are allowed to lend 'money' they don't have called 'credit. Go into a bank for a loan and if you succeed

the banker will not move any real wealth into your account. They will type into your account the amount of the agreed 'loan' – say £100,000. This is not wealth that really exists; it is non-existent, fresh-air, created-out-of-nothing 'credit' which has never, does not, and will never exist except in theory. Credit is backed by nothing except wind and only has buying power because people think that it has buying power and accept it in return for property, goods and services. I have described this situation as like those cartoon characters you see chasing each other and when they run over the edge of a cliff they keep running forward on fresh air until one of them looks down, realises what's happened, and they all crash into the ravine. The whole foundation of the Sabbatian financial system is to stop people looking down except for periodic moments when they want to crash the system (as in 2008 and 2020 ongoing) and reap the rewards from all the property, businesses and wealth their borrowers had signed over as 'collateral' in return for a 'loan' of fresh air. Most people think that money is somehow created by governments when it comes into existence from the start as a debt through banks 'lending' illusory money called credit. Yes, the very currency of exchange is a *debt* from day one issued as an interest-bearing loan. Why don't governments create money interest-free and lend it to their people interest-free? Governments are controlled by Sabbatians and the financial system is controlled by Sabbatians for whom interest-free money would be a nightmare come true. Sabbatians underpin their financial domination through their global network of central banks, including the privately-owned US Federal Reserve and Britain's Bank of England, and this is orchestrated by a privately-owned central bank coordination body called the Bank for International Settlements in Basle, Switzerland, created by the usual suspects including the Rockefellers and Rothschilds. Central bank chiefs don't answer to governments or the people. They answer to the Bank for International Settlements or, in other words, the Global Cult which is dominated today by Sabbatians.

Built-in disaster

There are so many constituent scams within the overall banking scam. When you take out a loan of thin-air credit only the amount of that loan is theoretically brought into circulation to add to the amount in circulation; but you are paying back the principle plus interest. The additional interest is not created and this means that with every 'loan' there is a shortfall in the money in circulation between what is borrowed and what has to be paid back. There is never even close to enough money in circulation to repay all outstanding public and private debt including interest. Coldly weaved in the very fabric of the system is the certainty that some will lose their homes, businesses and possessions to the banking 'lender'. This is less obvious in times of 'boom' when the amount of money in circulation (and the debt) is expanding through more people wanting and getting loans. When a downturn comes and the money supply contracts it becomes painfully obvious that there is not enough money to service all debt and interest. This is less obvious in times of 'boom' when the amount of money in circulation (and the debt) is expanding through more people wanting and getting loans. When a downturn comes and the money supply contracts and it becomes painfully obvious – as in 2008 and currently – that there is not enough money to service all debt and interest. Sabbatian banksters have been leading the human population through a calculated series of booms (more debt incurred) and busts (when the debt can't be repaid and the banks get the debtor's tangible wealth in exchange for non-existent 'credit'). With each 'bust' Sabbatian bankers have absorbed more of the world's tangible wealth and we end up with the One-percent. Governments are in bankruptcy levels of debt to the same system and are therefore owned by a system they do not control. The Federal Reserve, 'America's central bank', is privately-owned and American presidents only nominally appoint its chairman or woman to maintain the illusion that it's an arm of government. It's not. The 'Fed' is a cartel of private banks which handed billions to its associates and friends after the crash of 2008 and has been Sabbatian-controlled since it was manipulated into being in 1913 through the covert trickery of Rothschild banking agents Jacob Schiff and Paul

Warburg, and the Sabbatian Rockefeller family. Somehow from a Jewish population of two-percent and globally 0.2 percent (Sabbatian interlopers remember are far smaller) ultra-Zionists headed the Federal Reserve for 31 years between 1987 and 2018 in the form of Alan Greenspan, Bernard Bernanke and Janet Yellen (now Biden's Treasury Secretary) with Yellen's deputy chairman a Israeli-American duel citizen and ultra-Zionist Stanley Fischer, a former governor of the Bank of Israel. Ultra-Zionist Fed chiefs spanned the presidencies of Ronald Reagan ('Republican'), Father George Bush ('Republican'), Bill Clinton ('Democrat'), Boy George Bush ('Republican') and Barack Obama ('Democrat'). We should really add the pre-Greenspan chairman, Paul Adolph Volcker, 'appointed' by Jimmy Carter ('Democrat') who ran the Fed between 1979 and 1987 during the Carter and Reagan administrations before Greenspan took over. Volcker was a long-time associate and business partner of the Rothschilds. No matter what the 'party' officially in power the United States economy was directed by the same force. Here are members of the Obama, Trump and Biden administrations and see if you can make out a common theme.

Barack Obama ('Democrat')

Ultra-Zionists Robert Rubin, Larry Summers, and Timothy Geithner ran the US Treasury in the Clinton administration and two of them reappeared with Obama. Ultra-Zionist Fed chairman Alan Greenspan had manipulated the crash of 2008 through deregulation and jumped ship just before the disaster to make way for ultra-Zionist Bernard Bernanke to hand out trillions to Sabbatian 'too big to fail' banks and businesses, including the ubiquitous ultra-Zionist Goldman Sachs which has an ongoing staff revolving door operation between itself and major financial positions in government worldwide. Obama inherited the fallout of the crash when he took office in January, 2009, and fortunately he had the support of his ultra-Zionist White House Chief of Staff Rahm Emmanuel, son of a terrorist who helped to bomb Israel into being in 1948, and his ultra-Zionist senior adviser David Axelrod, chief strategist in Obama's two

successful presidential campaigns. Emmanuel, later mayor of Chicago and former senior fundraiser and strategist for Bill Clinton, is an example of the Sabbatian policy after Israel was established of migrating insider families to America so their children would be born American citizens. 'Obama' chose this financial team throughout his administration to respond to the Sabbatian-instigated crisis:

Timothy Geithner (ultra-Zionist) Treasury Secretary; Jacob J. Lew, Treasury Secretary; Larry Summers (ultra-Zionist), director of the White House National Economic Council; Paul Adolph Volcker (Rothschild business partner), chairman of the Economic Recovery Advisory Board; Peter Orszag (ultra-Zionist), director of the Office of Management and Budget overseeing all government spending; Penny Pritzker (ultra-Zionist), Commerce Secretary; Jared Bernstein (ultra-Zionist), chief economist and economic policy adviser to Vice President Joe Biden; Mary Schapiro (ultra-Zionist), chair of the Securities and Exchange Commission (SEC); Gary Gensler (ultra-Zionist), chairman of the Commodity Futures Trading Commission (CFTC); Sheila Bair (ultra-Zionist), chair of the Federal Deposit Insurance Corporation (FDIC); Karen Mills (ultra-Zionist), head of the Small Business Administration (SBA); Kenneth Feinberg (ultra-Zionist), Special Master for Executive [bail-out] Compensation. Feinberg would be appointed to oversee compensation (with strings) to 9/11 victims and families in a campaign to stop them having their day in court to question the official story. At the same time ultra-Zionist Bernard Bernanke was chairman of the Federal Reserve and these are only some of the ultra-Zionists with allegiance to Sabbatian-controlled Israel in the Obama government. Obama's biggest corporate donor was ultra-Zionist Goldman Sachs which had employed many in his administration.

Donald Trump ('Republican')

Trump claimed to be an outsider (he wasn't) who had come to 'drain the swamp'. He embarked on this goal by immediately appointing ultra-Zionist Steve Mnuchin, a Goldman Sachs employee for 17

years, as his Treasury Secretary. Others included Gary Cohn (ultra-Zionist), chief operating officer of Goldman Sachs, his first Director of the National Economic Council and chief economic adviser, who was later replaced by Larry Kudlow (ultra-Zionist). Trump's senior adviser throughout his four years in the White House was his sinister son-in-law Jared Kushner, a life-long friend of Israel Prime Minister Benjamin Netanyahu. Kushner is the son of a convicted crook who was pardoned by Trump in his last days in office. Other ultra-Zionists in the Trump administration included: Stephen Miller, Senior Policy Adviser; Avrahm Berkowitz, Deputy Adviser to Trump and his Senior Adviser Jared Kushner; Ivanka Trump, Adviser to the President, who converted to Judaism when she married Jared Kushner; David Friedman, Trump lawyer and Ambassador to Israel; Jason Greenblatt, Trump Organization executive vice president and chief legal officer, who was made Special Representative for International Negotiations and the Israeli-Palestinian Conflict; Rod Rosenstein, Deputy Attorney General; Elliot Abrams, Special Representative for Venezuela, then Iran; John Eisenberg, National Security Council Legal Adviser and Deputy Council to the President for National Security Affairs; Anne Neuberger, Deputy National Manager, National Security Agency; Ezra Cohen-Watnick, Acting Under Secretary of Defense for Intelligence; Elan Carr, Special Envoy to monitor and combat anti-Semitism; Len Khodorkovsky, Deputy Special Envoy to monitor and combat anti-Semitism; Reed Cordish, Assistant to the President, Intragovernmental and Technology Initiatives. Trump Vice President Mike Pence and Secretary of State Mike Pompeo, both Christian Zionists, were also vehement supporters of Israel and its goals and ambitions.

Donald 'free-speech believer' Trump pardoned a number of financial and violent criminals while ignoring calls to pardon Julian Assange and Edward Snowden whose crimes are revealing highly relevant information about government manipulation and corruption and the widespread illegal surveillance of the American people by US 'security' agencies. It's so good to know that Trump is on the side of freedom and justice and not mega-criminals with

allegiance to Sabbatian-controlled Israel. These included a pardon for Israeli spy Jonathan Pollard who was jailed for life in 1987 under the Espionage Act. Aviem Sella, the Mossad agent who recruited Pollard, was also pardoned by Trump while Assange sat in jail and Snowden remained in exile in Russia. Sella had 'fled' (was helped to escape) to Israel in 1987 and was never extradited despite being charged under the Espionage Act. A Trump White House statement said that Sella's clemency had been 'supported by Benjamin Netanyahu, Ron Dermer, Israel's US Ambassador, David Friedman, US Ambassador to Israel and Miriam Adelson, wife of leading Trump donor Sheldon Adelson who died shortly before. Other friends of Jared Kushner were pardoned along with Sholom Weiss who was believed to be serving the longest-ever white-collar prison sentence of more than 800 years in 2000. The sentence was commuted of Ponzi-schemer Eliyahu Weinstein who defrauded Jews and others out of $200 million. I did mention that Assange and Snowden were ignored, right? Trump gave Sabbatians almost everything they asked for in military and political support, moving the US Embassy from Tel Aviv to Jerusalem with its critical symbolic and literal implications for Palestinian statehood, and the 'deal of the Century' designed by Jared Kushner and David Friedman which gave the Sabbatian Israeli government the green light to substantially expand its already widespread program of building illegal Jewish-only settlements in the occupied land of the West Bank. This made a two-state 'solution' impossible by seizing all the land of a potential Palestinian homeland and that had been the plan since 1948 and then 1967 when the Arab-controlled Gaza Strip, West Bank, Sinai Peninsula and Syrian Golan Heights were occupied by Israel. All the talks about talks and road maps and delays have been buying time until the West Bank was physically occupied by Israeli real estate. Trump would have to be a monumentally ill-informed idiot not to see that this was the plan he was helping to complete. The Trump administration was in so many ways the Kushner administration which means the Netanyahu administration which means the Sabbatian administration. I understand why many opposing Cult fascism in all its forms gravitated to Trump, but he

was a crucial part of the Sabbatian plan and I will deal with this in the next chapter.

Joe Biden ('Democrat')

A barely cognitive Joe Biden took over the presidency in January, 2021, along with his fellow empty shell, Vice-President Kamala Harris, as the latest Sabbatian gofers to enter the White House. Names on the door may have changed and the 'party' – the force behind them remained the same as Zionists were appointed to a stream of pivotal areas relating to Sabbatian plans and policy. They included: Janet Yellen, Treasury Secretary, former head of the Federal Reserve, and still another ultra-Zionist running the US Treasury after Mnuchin (Trump), Lew and Geithner (Obama), and Summers and Rubin (Clinton); Anthony Blinken, Secretary of State; Wendy Sherman, Deputy Secretary of State (so that's 'Biden's' Sabbatian foreign policy sorted); Jeff Zients, White House coronavirus coordinator; Rochelle Walensky, head of the Centers for Disease Control; Rachel Levine, transgender deputy health secretary (that's 'Covid' hoax policy under control); Merrick Garland, Attorney General; Alejandro Mayorkas, Secretary of Homeland Security; Cass Sunstein, Homeland Security with responsibility for new immigration laws; Avril Haines, Director of National Intelligence; Anne Neuberger, National Security Agency cybersecurity director (note, cybersecurity); David Cohen, CIA Deputy Director; Ronald Klain, Biden's Chief of Staff (see Rahm Emanuel); Eric Lander, a 'leading geneticist', Office of Science and Technology Policy director (see Smart Grid, synthetic biology agenda); Jessica Rosenworcel, acting head of the Federal Communications Commission (FCC) which controls Smart Grid technology policy and electromagnetic communication systems including 5G. How can it be that so many pivotal positions are held by two-percent of the American population and 0.2 percent of the world population administration after administration no matter who is the president and what is the party? It's a coincidence? Of course it's not and this is why Sabbatians have built their colossal global web of interlocking 'anti-

hate' hate groups to condemn anyone who asks these glaring questions as an 'anti-Semite'. The way that Jewish people horrifically abused in Sabbatian-backed Nazi Germany are exploited to this end is stomach-turning and disgusting beyond words.

Political fusion

Sabbatian manipulation has reversed the roles of Republicans and Democrats and the same has happened in Britain with the Conservative and Labour Parties. Republicans and Conservatives were always labelled the 'right' and Democrats and Labour the 'left', but look at the policy positions now and the Democrat-Labour 'left' has moved further to the 'right' than Republicans and Conservatives under the banner of 'Woke', the Cult-created far-right tyranny. Where once the Democrat-Labour 'left' defended free speech and human rights they now seek to delete them and as I said earlier despite the 'Covid' fascism of the Jackboot Johnson Conservative government in the UK the Labour Party of leader Keir Starmer demanded even more extreme measures. The Labour Party has been very publicly absorbed by Sabbatians after a political and media onslaught against the previous leader, the weak and inept Jeremy Corbyn, over made-up allegations of 'anti-Semitism' both by him and his party. The plan was clear with this 'anti-Semite' propaganda and what was required in response was a swift and decisive 'fuck off' from Corbyn and a statement to expose the Anti-Semitism Industry (Sabbatian) attempt to silence Labour criticism of the Israeli government (Sabbatians) and purge the party of all dissent against the extremes of ultra-Zionism (Sabbatians). Instead Corbyn and his party fell to their knees and appeased the abusers which, by definition, is impossible. Appeasing one demand leads only to a new demand to be appeased until takeover is complete. Like I say – 'fuck off' would have been a much more effective policy and I have used it myself with great effect over the years when Sabbatians are on my case which is most of the time. I consider that fact a great compliment, by the way. The outcome of the Labour Party capitulation is that we now have a Sabbatian-controlled

Conservative Party 'opposed' by a Sabbatian-controlled Labour Party in a one-party Sabbatian state that hurtles towards the extremes of tyranny (the Sabbatian cult agenda). In America the situation is the same. Labour's Keir Starmer spends his days on his knees with his tongue out pointing to Tel Aviv, or I guess now Jerusalem, while Boris Johnson has an 'anti-Semitism czar' in the form of former Labour MP John Mann who keeps Starmer company on his prayer mat.

Sabbatian influence can be seen in Jewish members of the Labour Party who have been ejected for criticism of Israel including those from families that suffered in Nazi Germany. Sabbatians despise real Jewish people and target them even more harshly because it is so much more difficult to dub them 'anti-Semitic' although in their desperation they do try.

CHAPTER THREE

The Pushbacker sting

Until you realize how easy it is for your mind to be manipulated, you remain the puppet of someone else's game
Evita Ochel

I will use the presidencies of Trump and Biden to show how the manipulation of the one-party state plays out behind the illusion of political choice across the world. No two presidencies could – on the face of it – be more different and apparently at odds in terms of direction and policy.

A Renegade Mind sees beyond the obvious and focuses on outcomes and consequences and not image, words and waffle. The Cult embarked on a campaign to divide America between those who blindly support its agenda (the mentality known as 'Woke') and those who are pushing back on where the Cult and its Sabbatians want to go. This presents infinite possibilities for dividing and ruling the population by setting them at war with each other and allows a perceptual ring fence of demonisation to encircle the Pushbackers in a modern version of the Little Big Horn in 1876 when American cavalry led by Lieutenant Colonel George Custer were drawn into a trap, surrounded and killed by Native American tribes defending their land of thousands of years from being seized by the government. In this modern version the roles are reversed and it's those defending themselves from the Sabbatian government who are surrounded and the government that's seeking to destroy them. This trap was set years ago and to explain how we must return to 2016

and the emergence of Donald Trump as a candidate to be President of the United States. He set out to overcome the best part of 20 other candidates in the Republican Party before and during the primaries and was not considered by many in those early stages to have a prayer of living in the White House. The Republican Party was said to have great reservations about Trump and yet somehow he won the nomination. When you know how American politics works – politics in general – there is no way that Trump could have become the party's candidate unless the Sabbatian-controlled 'Neocons' that run the Republican Party wanted that to happen. We saw the proof in emails and documents made public by WikiLeaks that the Democratic Party hierarchy, or Democons, systematically undermined the campaign of Bernie Sanders to make sure that Sabbatian gofer Hillary Clinton won the nomination to be their presidential candidate. If the Democons could do that then the Neocons in the Republican Party could have derailed Trump in the same way. But they didn't and at that stage I began to conclude that Trump could well be the one chosen to be president. If that was the case the 'why' was pretty clear to see – the goal of dividing America between Cult agenda-supporting Wokers and Pushbackers who gravitated to Trump because he was telling them what they wanted to hear. His constituency of support had been increasingly ignored and voiceless for decades and profoundly through the eight years of Sabbatian puppet Barack Obama. Now here was someone speaking their language of pulling back from the incessant globalisation of political and economic power, the exporting of American jobs to China and elsewhere by 'American' (Sabbatian) corporations, the deletion of free speech, and the mass immigration policies that had further devastated job opportunities for the urban working class of all races and the once American heartlands of the Midwest.

Beware the forked tongue

Those people collectively sighed with relief that at last a political leader was apparently on their side, but another trait of the Renegade Mind is that you look even harder at people telling you

what you want to hear than those who are telling you otherwise. Obviously as I said earlier people wish what they want to hear to be true and genuine and they are much more likely to believe that than someone saying what they don't want to here and don't want to be true. Sales people are taught to be skilled in eliciting by calculated questioning what their customers want to hear and repeating that back to them as their own opinion to get their targets to like and trust them. Assets of the Cult are also sales people in the sense of selling perception. To read Cult manipulation you have to play the long and expanded game and not fall for the Vaudeville show of party politics. Both American parties are vehicles for the Cult and they exploit them in different ways depending on what the agenda requires at that moment. Trump and the Republicans were used to be the focus of dividing America and isolating Pushbackers to open the way for a Biden presidency to become the most extreme in American history by advancing the full-blown Woke (Cult) agenda with the aim of destroying and silencing Pushbackers now labelled Nazi Trump supporters and white supremacists.

Sabbatians wanted Trump in office for the reasons described by ultra-Zionist Saul Alinsky (1909-1972) who was promoting the Woke philosophy through 'community organising' long before anyone had heard of it. In those days it still went by its traditional name of Marxism. The reason for the manipulated Trump phenomenon was laid out in Alinsky's 1971 book, *Rules for Radicals*, which was his blueprint for overthrowing democratic and other regimes and replacing them with Sabbatian Marxism. Not surprisingly his to-do list was evident in the Sabbatian French and Russian 'Revolutions' and that in China which will become very relevant in the next chapter about the 'Covid' hoax. Among Alinsky's followers have been the deeply corrupt Barack Obama, House Speaker Nancy Pelosi and Hillary Clinton who described him as a 'hero'. All three are Sabbatian stooges with Pelosi personifying the arrogant corrupt idiocy that so widely fronts up for the Cult inner core. Predictably as a Sabbatian advocate of the 'light-bringer' Alinsky features Lucifer on the dedication page of his book as the original radical who gained

his own kingdom ('Earth' as we shall see). One of Alinsky's golden radical rules was to pick an individual and focus all attention, hatred and blame on them and not to target faceless bureaucracies and corporations. *Rules for Radicals* is really a Sabbatian handbook with its contents repeatedly employed all over the world for centuries and why wouldn't Sabbatians bring to power their designer-villain to be used as the individual on which all attention, hatred and blame was bestowed? This is what they did and the only question for me is how much Trump knew that and how much he was manipulated. A bit of both, I suspect. This was Alinsky's Trump technique from a man who died in 1972. The technique has spanned history:

> Pick the target, freeze it, personalize it, polarize it. Don't try to attack abstract corporations or bureaucracies. Identify a responsible individual. Ignore attempts to shift or spread the blame.

From the moment Trump came to illusory power everything was about him. It wasn't about Republican policy or opinion, but all about Trump. Everything he did was presented in negative, derogatory and abusive terms by the Sabbatian-dominated media led by Cult operations such as CNN, MSNBC, *The New York Times* and the Jeff Bezos-owned *Washington Post* – 'Pick the target, freeze it, personalize it, polarize it.' Trump was turned into a demon to be vilified by those who hated him and a demi-god loved by those who worshipped him. This, in turn, had his supporters, too, presented as equally demonic in preparation for the punchline later down the line when Biden was about to take office. It was here's a Trump, there's a Trump, everywhere a Trump, Trump. Virtually every news story or happening was filtered through the lens of 'The Donald'. You loved him or hated him and which one you chose was said to define you as Satan's spawn or a paragon of virtue. Even supporting some Trump policies or statements and not others was enough for an assault on your character. No shades of grey were or are allowed. Everything is black and white (literally and figuratively). A Californian I knew had her head utterly scrambled by her hatred for Trump while telling people they should love each other. She was so totally consumed by

Trump Derangement Syndrome as it became to be known that this glaring contradiction would never have occurred to her. By definition anyone who criticised Trump or praised his opponents was a hero and this lady described Joe Biden as 'a kind, honest gentleman' when he's a provable liar, mega-crook and vicious piece of work to boot. Sabbatians had indeed divided America using Trump as the fall-guy and all along the clock was ticking on the consequences for his supporters.

In hock to his masters

Trump gave Sabbatians via Israel almost everything they wanted in his four years. Ask and you shall receive was the dynamic between himself and Benjamin Netanyahu orchestrated by Trump's ultra-Zionist son-in-law Jared Kushner, his ultra-Zionist Ambassador to Israel, David Friedman, and ultra-Zionist 'Israel adviser', Jason Greenblatt. The last two were central to the running and protecting from collapse of his business empire, the Trump Organisation, and colossal business failures made him forever beholding to Sabbatian networks that bailed him out. By the start of the 1990s Trump owed $4 billion to banks that he couldn't pay and almost $1billion of that was down to him personally and not his companies. This mega-disaster was the result of building two new casinos in Atlantic City and buying the enormous Taj Mahal operation which led to crippling debt payments. He had borrowed fantastic sums from 72 banks with major Sabbatian connections and although the scale of debt should have had him living in a tent alongside the highway they never foreclosed. A plan was devised to lift Trump from the mire by BT Securities Corporation and Rothschild Inc. and the case was handled by Wilber Ross who had worked for the Rothschilds for 27 years. Ross would be named US Commerce Secretary after Trump's election. Another crucial figure in saving Trump was ultra-Zionist 'investor' Carl Icahn who bought the Taj Mahal casino. Icahn was made special economic adviser on financial regulation in the Trump administration. He didn't stay long but still managed to find time to make a tidy sum of a reported $31.3 million when he sold his

holdings affected by the price of steel three days before Trump imposed a 235 percent tariff on steel imports. What amazing bits of luck these people have. Trump and Sabbatian operatives have long had a close association and his mentor and legal adviser from the early 1970s until 1986 was the dark and genetically corrupt ultra-Zionist Roy Cohn who was chief counsel to Senator Joseph McCarthy's 'communist' witch-hunt in the 1950s. *Esquire* magazine published an article about Cohn with the headline 'Don't mess with Roy Cohn'. He was described as the most feared lawyer in New York and 'a ruthless master of dirty tricks ... [with] ... more than one Mafia Don on speed dial'. Cohn's influence, contacts, support and protection made Trump a front man for Sabbatians in New York with their connections to one of Cohn's many criminal employers, the 'Russian' Sabbatian Mafia. Israel-centric media mogul Rupert Murdoch was introduced to Trump by Cohn and they started a long friendship. Cohn died in 1986 weeks after being disbarred for unethical conduct by the Appellate Division of the New York State Supreme Court. The wheels of justice do indeed run slow given the length of Cohn's crooked career.

QAnon-sense

We are asked to believe that Donald Trump with his fundamental connections to Sabbatian networks and operatives has been leading the fight to stop the Sabbatian agenda for the fascistic control of America and the world. Sure he has. A man entrapped during his years in the White House by Sabbatian operatives and whose biggest financial donor was casino billionaire Sheldon Adelson who was Sabbatian to his DNA?? Oh, do come on. Trump has been used to divide America and isolate Pushbackers on the Cult agenda under the heading of 'Trump supporters', 'insurrectionists' and 'white supremacists'. The US Intelligence/Mossad Psyop or psychological operation known as QAnon emerged during the Trump years as a central pillar in the Sabbatian campaign to lead Pushbackers into the trap set by those that wished to destroy them. I knew from the start that QAnon was a scam because I had seen the same scenario many

times before over 30 years under different names and I had written about one in particular in the books. 'Not again' was my reaction when QAnon came to the fore. The same script is pulled out every few years and a new name added to the letterhead. The story always takes the same form: 'Insiders' or 'the good guys' in the government-intelligence-military 'Deep State' apparatus were going to instigate mass arrests of the 'bad guys' which would include the Rockefellers, Rothschilds, Barack Obama, Hillary Clinton, George Soros, etc., etc. Dates are given for when the 'good guys' are going to move in, but the dates pass without incident and new dates are given which pass without incident. The central message to Pushbackers in each case is that they don't have to do anything because there is 'a plan' and it is all going to be sorted by the 'good guys' on the inside. 'Trust the plan' was a QAnon mantra when the only plan was to misdirect Pushbackers into putting their trust in a Psyop they believed to be real. Beware, beware, those who tell you what you want to hear and always check it out. Right up to Biden's inauguration QAnon was still claiming that 'the Storm' was coming and Trump would stay on as president when Biden and his cronies were arrested and jailed. It was never going to happen and of course it didn't, but what did happen as a result provided that punchline to the Sabbatian Trump/QAnon Psyop.

On January 6th, 2021, a very big crowd of Trump supporters gathered in the National Mall in Washington DC down from the Capitol Building to protest at what they believed to be widespread corruption and vote fraud that stopped Trump being re-elected for a second term as president in November, 2020. I say as someone that does not support Trump or Biden that the evidence is clear that major vote-fixing went on to favour Biden, a man with cognitive problems so advanced he can often hardly string a sentence together without reading the words written for him on the Teleprompter. Glaring ballot discrepancies included serious questions about electronic voting machines that make vote rigging a comparative cinch and hundreds of thousands of paper votes that suddenly appeared during already advanced vote counts and virtually all of

them for Biden. Early Trump leads in crucial swing states suddenly began to close and disappear. The pandemic hoax was used as the excuse to issue almost limitless numbers of mail-in ballots with no checks to establish that the recipients were still alive or lived at that address. They were sent to streams of people who had not even asked for them. Private organisations were employed to gather these ballots and who knows what they did with them before they turned up at the counts. The American election system has been manipulated over decades to become a sick joke with more holes than a Swiss cheese for the express purpose of dictating the results. Then there was the criminal manipulation of information by Sabbatian tech giants like Facebook, Twitter and Google-owned YouTube which deleted pro-Trump, anti-Biden accounts and posts while everything in support of Biden was left alone. Sabbatians wanted Biden to win because after the dividing of America it was time for full-on Woke and every aspect of the Cult agenda to be unleashed.

Hunter gatherer

Extreme Silicon Valley bias included blocking information by the *New York Post* exposing a Biden scandal that should have ended his bid for president in the final weeks of the campaign. Hunter Biden, his monumentally corrupt son, is reported to have sent a laptop to be repaired at a local store and failed to return for it. Time passed until the laptop became the property of the store for non-payment of the bill. When the owner saw what was on the hard drive he gave a copy to the FBI who did nothing even though it confirmed widespread corruption in which the Joe Biden family were using his political position, especially when he was vice president to Obama, to make multiple millions in countries around the world and most notably Ukraine and China. Hunter Biden's one-time business partner Tony Bobulinski went public when the story broke in the *New York Post* to confirm the corruption he saw and that Joe Biden not only knew what was going on he also profited from the spoils. Millions were handed over by a Chinese company with close

connections – like all major businesses in China – to the Chinese communist party of President Xi Jinping. Joe Biden even boasted at a meeting of the Cult's World Economic Forum that as vice president he had ordered the government of Ukraine to fire a prosecutor. What he didn't mention was that the same man just happened to be investigating an energy company which was part of Hunter Biden's corrupt portfolio. The company was paying him big bucks for no other reason than the influence his father had. Overnight Biden's presidential campaign should have been over given that he had lied publicly about not knowing what his son was doing. Instead almost the entire Sabbatian-owned mainstream media and Sabbatian-owned Silicon Valley suppressed circulation of the story. This alone went a mighty way to rigging the election of 2020. Cult assets like Mark Zuckerberg at Facebook also spent hundreds of millions to be used in support of Biden and vote 'administration'.

The Cult had used Trump as the focus to divide America and was now desperate to bring in moronic, pliable, corrupt Biden to complete the double-whammy. No way were they going to let little things like the will of the people thwart their plan. Silicon Valley widely censored claims that the election was rigged because it *was* rigged. For the same reason anyone claiming it was rigged was denounced as a 'white supremacist' including the pathetically few Republican politicians willing to say so. Right across the media where the claim was mentioned it was described as a 'false claim' even though these excuses for 'journalists' would have done no research into the subject whatsoever. Trump won seven million more votes than any sitting president had ever achieved while somehow a cognitively-challenged soon to be 78-year-old who was hidden away from the public for most of the campaign managed to win more votes than any presidential candidate in history. It makes no sense. You only had to see election rallies for both candidates to witness the enthusiasm for Trump and the apathy for Biden. Tens of thousands would attend Trump events while Biden was speaking in empty car parks with often only television crews attending and framing their shots to hide the fact that no one was there. It was pathetic to see

footage come to light of Biden standing at a podium making speeches only to TV crews and party fixers while reading the words written for him on massive Teleprompter screens. So, yes, those protestors on January 6th had a point about election rigging, but some were about to walk into a trap laid for them in Washington by the Cult Deep State and its QAnon Psyop. This was the Capitol Hill riot ludicrously dubbed an 'insurrection'.

The spider and the fly

Renegade Minds know there are not two 'sides' in politics, only one side, the Cult, working through all 'sides'. It's a stage show, a puppet show, to direct the perceptions of the population into focusing on diversions like parties and candidates while missing the puppeteers with their hands holding all the strings. The Capitol Hill 'insurrection' brings us back to the Little Big Horn. Having created two distinct opposing groupings – Woke and Pushbackers – the trap was about to be sprung. Pushbackers were to be encircled and isolated by associating them all in the public mind with Trump and then labelling Trump as some sort of Confederate leader. I knew immediately that the Capitol riot was a set-up because of two things. One was how easy the rioters got into the building with virtually no credible resistance and secondly I could see – as with the 'Covid' hoax in the West at the start of 2020 – how the Cult could exploit the situation to move its agenda forward with great speed. My experience of Cult techniques and activities over more than 30 years has showed me that while they do exploit situations they haven't themselves created this never happens with events of fundamental agenda significance. Every time major events giving cultists the excuse to rapidly advance their plan you find they are manipulated into being for the specific reason of providing that excuse – Problem-Reaction-Solution. Only a tiny minority of the huge crowd of Washington protestors sought to gain entry to the Capitol by smashing windows and breaching doors. That didn't matter. The whole crowd and all Pushbackers, even if they did not support Trump, were going to be lumped together as dangerous

insurrectionists and conspiracy theorists. The latter term came into widespread use through a CIA memo in the 1960s aimed at discrediting those questioning the nonsensical official story of the Kennedy assassination and it subsequently became widely employed by the media. It's still being used by inept 'journalists' with no idea of its origin to discredit anyone questioning anything that authority claims to be true. When you are perpetrating a conspiracy you need to discredit the very word itself even though the dictionary definition of conspiracy is merely 'the activity of secretly planning with other people to do something bad or illegal' and 'a general agreement to keep silent about a subject for the purpose of keeping it secret'. On that basis there are conspiracies almost wherever you look. For obvious reasons the Cult and its lapdog media have to claim there are no conspiracies even though the word appears in state laws as with conspiracy to defraud, to murder, and to corrupt public morals.

Agent provocateurs are widely used by the Cult Deep State to manipulate genuine people into acting in ways that suit the desired outcome. By genuine in this case I mean protestors genuinely supporting Trump and claims that the election was stolen. In among them, however, were agents of the state wearing the garb of Trump supporters and QAnon to pump-prime the Capital riot which some genuine Trump supporters naively fell for. I described the situation as 'Come into my parlour said the spider to the fly'. Leaflets appeared through the Woke paramilitary arm Antifa, the anti-fascist fascists, calling on supporters to turn up in Washington looking like Trump supporters even though they hated him. Some of those arrested for breaching the Capitol Building were sourced to Antifa and its stable mate Black Lives Matter. Both organisations are funded by Cult billionaires and corporations. One man charged for the riot was according to his lawyer a former FBI agent who had held top secret security clearance for 40 years. Attorney Thomas Plofchan said of his client, 66-year-old Thomas Edward Caldwell:

He has held a Top Secret Security Clearance since 1979 and has undergone multiple Special Background Investigations in support of his clearances. After retiring from the Navy, he

worked as a section chief for the Federal Bureau of Investigation from 2009-2010 as a GS-12 [mid-level employee].

He also formed and operated a consulting firm performing work, often classified, for U.S government customers including the US. Drug Enforcement Agency, Department of Housing and Urban Development, the US Coast Guard, and the US Army Personnel Command.

A judge later released Caldwell pending trial in the absence of evidence about a conspiracy or that he tried to force his way into the building. *The New York Post* reported a 'law enforcement source' as saying that 'at least two known Antifa members were spotted' on camera among Trump supporters during the riot while one of the rioters arrested was John Earle Sullivan, a seriously extreme Black Lives Matter Trump-hater from Utah who was previously arrested and charged in July, 2020, over a BLM-Antifa riot in which drivers were threatened and one was shot. Sullivan is the founder of Utah-based Insurgence USA which is an affiliate of the Cult-created-and-funded Black Lives Matter movement. Footage appeared and was then deleted by Twitter of Trump supporters calling out Antifa infiltrators and a group was filmed changing into pro-Trump clothing before the riot. Security at the building was *pathetic* – as planned. Colonel Leroy Fletcher Prouty, a man with long experience in covert operations working with the US security apparatus, once described the tell-tale sign to identify who is involved in an assassination. He said:

No one has to direct an assassination – it happens. The active role is played secretly by permitting it to happen. This is the greatest single clue. Who has the power to call off or reduce the usual security precautions?

This principle applies to many other situations and certainly to the Capitol riot of January 6th, 2021.

The sting

With such a big and potentially angry crowd known to be gathering near the Capitol the security apparatus would have had a major police detail to defend the building with National Guard troops on

standby given the strength of feeling among people arriving from all over America encouraged by the QAnon Psyop and statements by Donald Trump. Instead Capitol Police 'security' was flimsy, weak, and easily breached. The same number of officers was deployed as on a regular day and that is a blatant red flag. They were not staffed or equipped for a possible riot that had been an obvious possibility in the circumstances. No protective and effective fencing worth the name was put in place and there were no contingency plans. The whole thing was basically a case of standing aside and waving people in. Once inside police mostly backed off apart from one Capitol police officer who ridiculously shot dead unarmed Air Force veteran protestor Ashli Babbitt without a warning as she climbed through a broken window. The 'investigation' refused to name or charge the officer after what must surely be considered a murder in the circumstances. They just lifted a carpet and swept. The story was endlessly repeated about five people dying in the 'armed insurrection' when there was no report of rioters using weapons. Apart from Babbitt the other four died from a heart attack, strokes and apparently a drug overdose. Capitol police officer Brian Sicknick was reported to have died after being bludgeoned with a fire extinguisher when he was alive after the riot was over and died later of what the Washington Medical Examiner's Office said was a stroke. Sicknick had no external injuries. The lies were delivered like rapid fire. There was a narrative to build with incessant repetition of the lie until the lie became the accepted 'everybody knows that' truth. The 'Big Lie' technique of Nazi Propaganda Minister Joseph Goebbels is constantly used by the Cult which was behind the Nazis and is today behind the 'Covid' and 'climate change' hoaxes. Goebbels said:

> If you tell a lie big enough and keep repeating it, people will eventually come to believe it. The lie can be maintained only for such time as the State can shield the people from the political, economic and/or military consequences of the lie. It thus becomes vitally important for the State to use all of its powers to repress dissent, for the truth is the mortal enemy of the lie, and thus by extension, the truth is the greatest enemy of the State.

Most protestors had a free run of the Capitol Building. This allowed pictures to be taken of rioters in iconic parts of the building including the Senate chamber which could be used as propaganda images against all Pushbackers. One Congresswoman described the scene as 'the worst kind of non-security anybody could ever imagine'. Well, the first part was true, but someone obviously did imagine it and made sure it happened. Some photographs most widely circulated featured people wearing QAnon symbols and now the Psyop would be used to dub all QAnon followers with the ubiquitous fit-all label of 'white supremacist' and 'insurrectionists'. When a Muslim extremist called Noah Green drove his car at two police officers at the Capitol Building killing one in April, 2021, there was no such political and media hysteria. They were just disappointed he wasn't white.

The witch-hunt

Government prosecutor Michael Sherwin, an aggressive, dark-eyed, professional Rottweiler led the 'investigation' and to call it over the top would be to understate reality a thousand fold. Hundreds were tracked down and arrested for the crime of having the wrong political views and people were jailed who had done nothing more than walk in the building, committed no violence or damage to property, took a few pictures and left. They were labelled a 'threat to the Republic' while Biden sat in the White House signing executive orders written for him that were dismantling 'the Republic'. Even when judges ruled that a mother and son should not be in jail the government kept them there. Some of those arrested have been badly beaten by prison guards in Washington and lawyers for one man said he suffered a fractured skull and was made blind in one eye. Meanwhile a woman is shot dead for no reason by a Capitol Police officer and we are not allowed to know who he is never mind what has happened to him although that will be *nothing*. The Cult's QAnon/Trump sting to identify and isolate Pushbackers and then target them on the road to crushing and deleting them was a resounding success. You would have thought the Russians had

invaded the building at gunpoint and lined up senators for a firing squad to see the political and media reaction. Congresswoman Alexandria Ocasio-Cortez is a child in a woman's body, a terrible-twos, me, me, me, Woker narcissist of such proportions that words have no meaning. She said she thought she was going to die when 'insurrectionists' banged on her office door. It turned out she wasn't even in the Capitol Building when the riot was happening and the 'banging' was a Capitol Police officer. She referred to herself as a 'survivor' which is an insult to all those true survivors of violent and sexual abuse while she lives her pampered and privileged life talking drivel for a living. Her Woke colleague and fellow mega-narcissist Rashida Tlaib broke down describing the devastating effect on her, too, of *not being* in the building when the rioters were there. Ocasio-Cortez and Tlaib are members of a fully-Woke group of Congresswomen known as 'The Squad' along with Ilhan Omar and Ayanna Pressley. The Squad from what I can see can be identified by its vehement anti-white racism, anti-white men agenda, and, as always in these cases, the absence of brain cells on active duty.

The usual suspects were on the riot case immediately in the form of Democrat ultra-Zionist senators and operatives Chuck Schumer and Adam Schiff demanding that Trump be impeached for 'his part in the insurrection'. The same pair of prats had led the failed impeachment of Trump over the invented 'Russia collusion' nonsense which claimed Russia had helped Trump win the 2016 election. I didn't realise that Tel Aviv had been relocated just outside Moscow. I must find an up-to-date map. The Russia hoax was a Sabbatian operation to keep Trump occupied and impotent and to stop any rapport with Russia which the Cult wants to retain as a perceptual enemy to be pulled out at will. Puppet Biden began attacking Russia when he came to office as the Cult seeks more upheaval, division and war across the world. A two-year stage show 'Russia collusion inquiry' headed by the not-very-bright former 9/11 FBI chief Robert Mueller, with support from 19 lawyers, 40 FBI agents plus intelligence analysts, forensic accountants and other

staff, devoured tens of millions of dollars and found no evidence of Russia collusion which a ten-year-old could have told them on day one. Now the same moronic Schumer and Schiff wanted a second impeachment of Trump over the Capitol 'insurrection' (riot) which the arrested development of Schumer called another 'Pearl Harbor' while others compared it with 9/11 in which 3,000 died and, in the case of CNN, with the Rwandan genocide in the 1990s in which an estimated 500,000 to 600,000 were murdered, between 250, 000 and 500,000 women were raped, and populations of whole towns were hacked to death with machetes. To make those comparisons purely for Cult political reasons is beyond insulting to those that suffered and lost their lives and confirms yet again the callous inhumanity that we are dealing with. Schumer is a monumental idiot and so is Schiff, but they serve the Cult agenda and do whatever they're told so they get looked after. Talking of idiots – another inane man who spanned the Russia and Capitol impeachment attempts was Senator Eric Swalwell who had the nerve to accuse Trump of collusion with the Russians while sleeping with a Chinese spy called Christine Fang or 'Fang Fang' which is straight out of a Bond film no doubt starring Klaus Schwab as the bloke living on a secret island and controlling laser weapons positioned in space and pointing at world capitals. Fang Fang plays the part of Bond's infiltrator girlfriend which I'm sure she would enjoy rather more than sharing a bed with the brainless Swalwell, lying back and thinking of China. The FBI eventually warned Swalwell about Fang Fang which gave her time to escape back to the Chinese dictatorship. How very thoughtful of them. The second Trump impeachment also failed and hardly surprising when an impeachment is supposed to remove a sitting president and by the time it happened Trump was no longer president. These people are running your country America, well, officially anyway. Terrifying isn't it?

Outcomes tell the story - always

The outcome of all this – and it's the *outcome* on which Renegade Minds focus, not the words – was that a vicious, hysterical and

obviously pre-planned assault was launched on Pushbackers to censor, silence and discredit them and even targeted their right to earn a living. They have since been condemned as 'domestic terrorists' that need to be treated like Al-Qaeda and Islamic State. 'Domestic terrorists' is a label the Cult has been trying to make stick since the period of the Oklahoma bombing in 1995 which was blamed on 'far-right domestic terrorists'. If you read *The Trigger* you will see that the bombing was clearly a Problem-Reaction-Solution carried out by the Deep State during a Bill Clinton administration so corrupt that no dictionary definition of the term would even nearly suffice. Nearly 30, 000 troops were deployed from all over America to the empty streets of Washington for Biden's inauguration. Ten thousand of them stayed on with the pretext of protecting the capital from insurrectionists when it was more psychological programming to normalise the use of the military in domestic law enforcement in support of the Cult plan for a police-military state. Biden's fascist administration began a purge of 'wrong-thinkers' in the military which means anyone that is not on board with Woke. The Capitol Building was surrounded by a fence with razor wire and the Land of the Free was further symbolically and literally dismantled. The circle was completed with the installation of Biden and the exploitation of the QAnon Psyop.

America had never been so divided since the civil war of the 19th century, Pushbackers were isolated and dubbed terrorists and now, as was always going to happen, the Cult immediately set about deleting what little was left of freedom and transforming American society through a swish of the hand of the most controlled 'president' in American history leading (officially at least) the most extreme regime since the country was declared an independent state on July 4th, 1776. Biden issued undebated, dictatorial executive orders almost by the hour in his opening days in office across the whole spectrum of the Cult wish-list including diluting controls on the border with Mexico allowing thousands of migrants to illegally enter the United States to transform the demographics of America and import an election-changing number of perceived Democrat

voters. Then there were Biden deportation amnesties for the already illegally resident (estimated to be as high as 20 or even 30 million). A bill before Congress awarded American citizenship to anyone who could prove they had worked in agriculture for just 180 days in the previous two years as 'Big Ag' secured its slave labour long-term. There were the plans to add new states to the union such as Puerto Rico and making Washington DC a state. They are all parts of a plan to ensure that the Cult-owned Woke Democrats would be permanently in power.

Border – what border?

I have exposed in detail in other books how mass immigration into the United States and Europe is the work of Cult networks fuelled by the tens of billions spent to this and other ends by George Soros and his global Open Society (open borders) Foundations. The impact can be seen in America alone where the population has increased by *100 million* in little more than 30 years mostly through immigration. I wrote in *The Answer* that the plan was to have so many people crossing the southern border that the numbers become unstoppable and we are now there under Cult-owned Biden. El Salvador in Central America puts the scale of what is happening into context. A third of the population now lives in the United States, much of it illegally, and many more are on the way. The methodology is to crush Central and South American countries economically and spread violence through machete-wielding psychopathic gangs like MS-13 based in El Salvador and now operating in many American cities. Biden-imposed lax security at the southern border means that it is all but open. He said before his 'election' that he wanted to see a surge towards the border if he became president and that was the green light for people to do just that after election day to create the human disaster that followed for both America and the migrants. When that surge came the imbecilic Alexandria Ocasio-Cortez said it wasn't a 'surge' because they are 'children, not insurgents' and the term 'surge' (used by Biden) was a claim of 'white supremacists'.

This disingenuous lady may one day enter the realm of the most basic intelligence, but it won't be any time soon.

Sabbatians and the Cult are in the process of destroying America by importing violent people and gangs in among the genuine to terrorise American cities and by overwhelming services that cannot cope with the sheer volume of new arrivals. Something similar is happening in Europe as Western society in general is targeted for demographic and cultural transformation and upheaval. The plan demands violence and crime to create an environment of intimidation, fear and division and Soros has been funding the election of district attorneys across America who then stop prosecuting many crimes, reduce sentences for violent crimes and free as many violent criminals as they can. Sabbatians are creating the chaos from which order – their order – can respond in a classic Problem-Reaction-Solution. A Freemasonic moto says 'Ordo Ab Chao' (Order out of Chaos) and this is why the Cult is constantly creating chaos to impose a new 'order'. Here you have the reason the Cult is constantly creating chaos. The 'Covid' hoax can be seen with those entering the United States by plane being forced to take a 'Covid' test while migrants flooding through southern border processing facilities do not. Nothing is put in the way of mass migration and if that means ignoring the government's own 'Covid' rules then so be it. They know it's all bullshit anyway. Any pushback on this is denounced as 'racist' by Wokers and Sabbatian fronts like the ultra-Zionist Anti-Defamation League headed by the appalling Jonathan Greenblatt which at the same time argues that Israel should not give citizenship and voting rights to more Palestinian Arabs or the 'Jewish population' (in truth the Sabbatian network) will lose control of the country.

Society-changing numbers

Biden's masters have declared that countries like El Salvador are so dangerous that their people must be allowed into the United States for humanitarian reasons when there are fewer murders in large parts of many Central American countries than in US cities like

Baltimore. That is not to say Central America cannot be a dangerous place and Cult-controlled American governments have been making it so since way back, along with the dismantling of economies, in a long-term plan to drive people north into the United States. Parts of Central America are very dangerous, but in other areas the story is being greatly exaggerated to justify relaxing immigration criteria. Migrants are being offered free healthcare and education in the United States as another incentive to head for the border and there is no requirement to be financially independent before you can enter to prevent the resources of America being drained. You can't blame migrants for seeking what they believe will be a better life, but they are being played by the Cult for dark and nefarious ends. The numbers since Biden took office are huge. In February, 2021, more than 100,000 people were known to have tried to enter the US illegally through the southern border (it was 34,000 in the same month in 2020) and in March it was 170,000 – a 418 percent increase on March, 2020. These numbers are only known people, not the ones who get in unseen. The true figure for migrants illegally crossing the border in a single month was estimated by one congressman at 250,000 and that number will only rise under Biden's current policy. Gangs of murdering drug-running thugs that control the Mexican side of the border demand money – thousands of dollars – to let migrants cross the Rio Grande into America. At the same time gun battles are breaking out on the border several times a week between rival Mexican drug gangs (which now operate globally) who are equipped with sophisticated military-grade weapons, grenades and armoured vehicles. While the Capitol Building was being 'protected' from a non-existent 'threat' by thousands of troops, and others were still deployed at the time in the Cult Neocon war in Afghanistan, the southern border of America was left to its fate. This is not incompetence, it is cold calculation.

By March, 2021, there were 17,000 unaccompanied children held at border facilities and many of them are ensnared by people traffickers for paedophile rings and raped on their journey north to America. This is not conjecture – this is fact. Many of those designated

children are in reality teenage boys or older. Meanwhile Wokers posture their self-purity for encouraging poor and tragic people to come to America and face this nightmare both on the journey and at the border with the disgusting figure of House Speaker Nancy Pelosi giving disingenuous speeches about caring for migrants. The woman's evil. Wokers condemned Trump for having children in cages at the border (so did Obama, *Shhhh*), but now they are sleeping on the floor without access to a shower with one border facility 729 percent over capacity. The Biden insanity even proposed flying migrants from the southern border to the northern border with Canada for 'processing'. The whole shambles is being overseen by ultra-Zionist Secretary of Homeland Security, the moronic liar Alejandro Mayorkas, who banned news cameras at border facilities to stop Americans seeing what was happening. Mayorkas said there was not a ban on news crews; it was just that they were not allowed to film. Alongside him at Homeland Security is another ultra-Zionist Cass Sunstein appointed by Biden to oversee new immigration laws. Sunstein despises conspiracy researchers to the point where he suggests they should be banned or *taxed* for having such views. The man is not bonkers or anything. He's perfectly well-adjusted, but adjusted to what is the question. Criticise what is happening and you are a 'white supremacist' when earlier non-white immigrants also oppose the numbers which effect their lives and opportunities. Black people in poor areas are particularly damaged by uncontrolled immigration and the increased competition for work opportunities with those who will work for less. They are also losing voting power as Hispanics become more dominant in former black areas. It's a downward spiral for them while the billionaires behind the policy drone on about how much they care about black people and 'racism'. None of this is about compassion for migrants or black people – that's just wind and air. Migrants are instead being mercilessly exploited to transform America while the countries they leave are losing their future and the same is true in Europe. Mass immigration may now be the work of Woke Democrats, but it can be traced back to the 1986 Immigration Reform and Control Act (it

wasn't) signed into law by Republican hero President Ronald Reagan which gave amnesty to millions living in the United States illegally and other incentives for people to head for the southern border. Here we have the one-party state at work again.

Save me syndrome

Almost every aspect of what I have been exposing as the Cult agenda was on display in even the first days of 'Biden' with silencing of Pushbackers at the forefront of everything. A Renegade Mind will view the Trump years and QAnon in a very different light to their supporters and advocates as the dots are connected. The QAnon/Trump Psyop has given the Cult all it was looking for. We may not know how much, or little, that Trump realised he was being used, but that's a side issue. This pincer movement produced the desired outcome of dividing America and having Pushbackers isolated. To turn this around we have to look at new routes to empowerment which do not include handing our power to other people and groups through what I will call the 'Save Me Syndrome' – 'I want someone else to do it so that I don't have to'. We have seen this at work throughout human history and the QAnon/Trump Psyop is only the latest incarnation alongside all the others. Religion is an obvious expression of this when people look to a 'god' or priest to save them or tell them how to be saved and then there are 'save me' politicians like Trump. Politics is a diversion and not a 'saviour'. It is a means to block positive change, not make it possible.

Save Me Syndrome always comes with the same repeating theme of handing your power to whom or what you believe will save you while your real 'saviour' stares back from the mirror every morning. Renegade Minds are constantly vigilant in this regard and always asking the question 'What can I do?' rather than 'What can someone else do for me?' Gandhi was right when he said: 'You must be the change you want to see in the world.' We are indeed the people we have been waiting for. We are presented with a constant raft of reasons to concede that power to others and forget where the real power is. Humanity has the numbers and the Cult does not. It has to

use diversion and division to target the unstoppable power that comes from unity. Religions, governments, politicians, corporations, media, QAnon, are all different manifestations of this power-diversion and dilution. Refusing to give your power to governments and instead handing it to Trump and QAnon is not to take a new direction, but merely to recycle the old one with new names on the posters. I will explore this phenomenon as we proceed and how to break the cycles and recycles that got us here through the mists of repeating perception and so repeating history.

For now we shall turn to the most potent example in the entire human story of the consequences that follow when you give your power away. I am talking, of course, of the 'Covid' hoax.

CHAPTER FOUR

'Covid': Calculated catastrophe

Facts are threatening to those invested in fraud
DaShanne Stokes

We can easily unravel the real reason for the 'Covid pandemic' hoax by employing the Renegade Mind methodology that I have outlined this far. We'll start by comparing the long-planned Cult outcome with the 'Covid pandemic' outcome. Know the outcome and you'll see the journey.

I have highlighted the plan for the Hunger Games Society which has been in my books for so many years with the very few controlling the very many through ongoing dependency. To create this dependency it is essential to destroy independent livelihoods, businesses and employment to make the population reliant on the state (the Cult) for even the basics of life through a guaranteed pittance income. While independence of income remained these Cult ambitions would be thwarted. With this knowledge it was easy to see where the 'pandemic' hoax was going once talk of 'lockdowns' began and the closing of all but perceived 'essential' businesses to 'save' us from an alleged 'deadly virus'. Cult corporations like Amazon and Walmart were naturally considered 'essential' while mom and pop shops and stores had their doors closed by fascist decree. As a result with every new lockdown and new regulation more small and medium, even large businesses not owned by the Cult, went to the wall while Cult giants and their frontmen and women grew financially fatter by the second. Mom and pop were

denied an income and the right to earn a living and the wealth of people like Jeff Bezos (Amazon), Mark Zuckerberg (Facebook) and Sergei Brin and Larry Page (Google/Alphabet) have reached record levels. The Cult was increasing its own power through further dramatic concentrations of wealth while the competition was being destroyed and brought into a state of dependency. Lockdowns have been instigated to secure that very end and were never anything to do with health. My brother Paul spent 45 years building up a bus repair business, but lockdowns meant buses were running at a fraction of normal levels for months on end. Similar stories can told in their hundreds of millions worldwide. Efforts of a lifetime coldly destroyed by Cult multi-billionaires and their lackeys in government and law enforcement who continued to earn their living from the taxation of the people while denying the right of the same people to earn theirs. How different it would have been if those making and enforcing these decisions had to face the same financial hardships of those they affected, but they never do.

Gates of Hell

Behind it all in the full knowledge of what he is doing and why is the psychopathic figure of Cult operative Bill Gates. His puppet Tedros at the World Health Organization declared 'Covid' a pandemic in March, 2020. The WHO had changed the definition of a 'pandemic' in 2009 just a month before declaring the 'swine flu pandemic' which would not have been so under the previous definition. The same applies to 'Covid'. The definition had included… 'an infection by an infectious agent, occurring simultaneously in different countries, with a significant mortality rate relative to the proportion of the population infected'. The new definition removed the need for 'significant mortality'. The 'pandemic' has been fraudulent even down to the definition, but Gates demanded economy-destroying lockdowns, school closures, social distancing, mandatory masks, a 'vaccination' for every man, woman and child on the planet and severe consequences and restrictions for those that refused. Who gave him this power? The

Cult did which he serves like a little boy in short trousers doing what his daddy tells him. He and his psychopathic missus even smiled when they said that much worse was to come (what they knew was planned to come). Gates responded in the matter-of-fact way of all psychopaths to a question about the effect on the world economy of what he was doing:

> Well, it won't go to zero but it will shrink. Global GDP is probably going to take the biggest hit ever [Gates was smiling as he said this] … in my lifetime this will be the greatest economic hit. But you don't have a choice. People act as if you have a choice. People don't feel like going to the stadium when they might get infected … People are deeply affected by seeing these stats, by knowing they could be part of the transmission chain, old people, their parents and grandparents, could be affected by this, and so you don't get to say ignore what is going on here.
>
> There will be the ability to open up, particularly in rich countries, if things are done well over the next few months, but for the world at large normalcy only returns when we have largely vaccinated the entire population.

The man has no compassion or empathy. How could he when he's a psychopath like all Cult players? My own view is that even beyond that he is very seriously mentally ill. Look in his eyes and you can see this along with his crazy flailing arms. You don't do what he has done to the world population since the start of 2020 unless you are mentally ill and at the most extreme end of psychopathic. You especially don't do it when to you know, as we shall see, that cases and deaths from 'Covid' are fakery and a product of monumental figure massaging. 'These stats' that Gates referred to are based on a 'test' that's not testing for the 'virus' as he has known all along. He made his fortune with big Cult support as an infamously ruthless software salesman and now buys global control of 'health' (death) policy without the population he affects having any say. It's a breathtaking outrage. Gates talked about people being deeply affected by fear of 'Covid' when that was because of *him* and his global network lying to them minute-by-minute supported by a lying media that he seriously influences and funds to the tune of hundreds of millions. He's handed big sums to media operations including the BBC, NBC, Al Jazeera, Univision, *PBS NewsHour*,

ProPublica, National Journal, The Guardian, The Financial Times, The Atlantic, Texas Tribune, USA Today publisher Gannett, *Washington Monthly, Le Monde,* Center for Investigative Reporting, Pulitzer Center on Crisis Reporting, National Press Foundation, International Center for Journalists, Solutions Journalism Network, the Poynter Institute for Media Studies, and many more. Gates is everywhere in the 'Covid' hoax and the man must go to prison – or a mental facility – for the rest of his life and his money distributed to those he has taken such enormous psychopathic pleasure in crushing.

The Muscle

The Hunger Games global structure demands a police-military state – a fusion of the two into one force – which viciously imposes the will of the Cult on the population and protects the Cult from public rebellion. In that regard, too, the 'Covid' hoax just keeps on giving. Often unlawful, ridiculous and contradictory 'Covid' rules and regulations have been policed across the world by moronic automatons and psychopaths made faceless by face-nappy masks and acting like the Nazi SS and fascist blackshirts and brownshirts of Hitler and Mussolini. The smallest departure from the rules decreed by the psychos in government and their clueless gofers were jumped upon by the face-nappy fascists. Brutality against public protestors soon became commonplace even on girls, women and old people as the brave men with the batons – the Face-Nappies as I call them – broke up peaceful protests and handed out fines like confetti to people who couldn't earn a living let alone pay hundreds of pounds for what was once an accepted human right. Robot Face-Nappies of Nottingham police in the English East Midlands fined one group £11,000 for attending a child's birthday party. For decades I charted the transformation of law enforcement as genuine, decent officers were replaced with psychopaths and the brain dead who would happily and brutally do whatever their masters told them. Now they were let loose on the public and I would emphasise the point that none of this just happened. The step-by-step change in the dynamic between police and public was orchestrated from the shadows by

those who knew where this was all going and the same with the perceptual reframing of those in all levels of authority and official administration through 'training courses' by organisations such as Common Purpose which was created in the late 1980s and given a massive boost in Blair era Britain until it became a global phenomenon. Supposed public 'servants' began to view the population as the enemy and the same was true of the police. This was the start of the explosion of behaviour manipulation organisations and networks preparing for the all-war on the human psyche unleashed with the dawn of 2020. I will go into more detail about this later in the book because it is a core part of what is happening.

Police desecrated beauty spots to deter people gathering and arrested women for walking in the countryside alone 'too far' from their homes. We had arrogant, clueless sergeants in the Isle of Wight police where I live posting on Facebook what they insisted the population must do or else. A schoolmaster sergeant called Radford looked young enough for me to ask if his mother knew he was out, but he was posting what he *expected* people to do while a Sergeant Wilkinson boasted about fining lads for meeting in a McDonald's car park where they went to get a lockdown takeaway. Wilkinson added that he had even cancelled their order. What a pair of prats these people are and yet they have increasingly become the norm among Jackboot Johnson's Yellowshirts once known as the British police. This was the theme all over the world with police savagery common during lockdown protests in the United States, the Netherlands, and the fascist state of Victoria in Australia under its tyrannical and again moronic premier Daniel Andrews. Amazing how tyrannical and moronic tend to work as a team and the same combination could be seen across America as arrogant, narcissistic Woke governors and mayors such as Gavin Newsom (California), Andrew Cuomo (New York), Gretchen Whitmer (Michigan), Lori Lightfoot (Chicago) and Eric Garcetti (Los Angeles) did their Nazi and Stalin impressions with the full support of the compliant brutality of their enforcers in uniform as they arrested small business owners defying

fascist shutdown orders and took them to jail in ankle shackles and handcuffs. This happened to bistro owner Marlena Pavlos-Hackney in Gretchen Whitmer's fascist state of Michigan when police arrived to enforce an order by a state-owned judge for 'putting the community at risk' at a time when other states like Texas were dropping restrictions and migrants were pouring across the southern border without any 'Covid' questions at all. I'm sure there are many officers appalled by what they are ordered to do, but not nearly enough of them. If they were truly appalled they would not do it. As the months passed every opportunity was taken to have the military involved to make their presence on the streets ever more familiar and 'normal' for the longer-term goal of police-military fusion.

Another crucial element to the Hunger Games enforcement network has been encouraging the public to report neighbours and others for 'breaking the lockdown rules'. The group faced with £11,000 in fines at the child's birthday party would have been dobbed-in by a neighbour with a brain the size of a pea. The technique was most famously employed by the Stasi secret police in communist East Germany who had public informants placed throughout the population. A police chief in the UK says his force doesn't need to carry out 'Covid' patrols when they are flooded with so many calls from the public reporting other people for visiting the beach. Dorset police chief James Vaughan said people were so enthusiastic about snitching on their fellow humans they were now operating as an auxiliary arm of the police: 'We are still getting around 400 reports a week from the public, so we will respond to reports …We won't need to be doing hotspot patrols because people are very quick to pick the phone up and tell us.' Vaughan didn't say that this is a pillar of all tyrannies of whatever complexion and the means to hugely extend the reach of enforcement while spreading distrust among the people and making them wary of doing anything that might get them reported. Those narcissistic Isle of Wight sergeants Radford and Wilkinson never fail to add a link to their Facebook posts where the public can inform on their fellow slaves.

Neither would be self-aware enough to realise they were imitating the Stasi which they might well never have heard of. Government psychologists that I will expose later laid out a policy to turn communities against each other in the same way.

A coincidence? Yep, and I can knit fog

I knew from the start of the alleged pandemic that this was a Cult operation. It presented limitless potential to rapidly advance the Cult agenda and exploit manipulated fear to demand that every man, woman and child on the planet was 'vaccinated' in a process never used on humans before which infuses self-replicating *synthetic* material into human cells. Remember the plan to transform the human body from a biological to a synthetic biological state. I'll deal with the 'vaccine' (that's not actually a vaccine) when I focus on the genetic agenda. Enough to say here that mass global 'vaccination' justified by this 'new virus' set alarms ringing after 30 years of tracking these people and their methods. The 'Covid' hoax officially beginning in China was also a big red flag for reasons I will be explaining. The agenda potential was so enormous that I could dismiss any idea that the 'virus' appeared naturally. Major happenings with major agenda implications never occur without Cult involvement in making them happen. My questions were twofold in early 2020 as the media began its campaign to induce global fear and hysteria: Was this alleged infectious agent released on purpose by the Cult or did it even exist at all? I then did what I always do in these situations. I sat, observed and waited to see where the evidence and information would take me. By March and early April synchronicity was strongly – and ever more so since then – pointing me in the direction of *there is no 'virus'*. I went public on that with derision even from swathes of the alternative media that voiced a scenario that the Chinese government released the 'virus' in league with Deep State elements in the United States from a top-level bio-lab in Wuhan where the 'virus' is said to have first appeared. I looked at that possibility, but I didn't buy it for several reasons. Deaths from the 'virus' did not in any way match what they

would have been with a 'deadly bioweapon' and it is much more effective if you sell the *illusion* of an infectious agent rather than having a real one unless you can control through injection who has it and who doesn't. Otherwise you lose control of events. A made-up 'virus' gives you a blank sheet of paper on which you can make it do whatever you like and have any symptoms or mutant 'variants' you choose to add while a real infectious agent would limit you to what it actually does. A phantom disease allows you to have endless ludicrous 'studies' on the 'Covid' dollar to widen the perceived impact by inventing ever more 'at risk' groups including one study which said those who walk slowly may be almost four times more likely to die from the 'virus'. People are in psychiatric wards for less.

A real 'deadly bioweapon' can take out people in the hierarchy that are not part of the Cult, but essential to its operation. Obviously they don't want that. Releasing a real disease means you immediately lose control of it. Releasing an illusory one means you don't. Again it's vital that people are extra careful when dealing with what they want to hear. A bioweapon unleashed from a Chinese laboratory in collusion with the American Deep State may fit a conspiracy narrative, but is it true? Would it not be far more effective to use the excuse of a 'virus' to justify the real bioweapon – the 'vaccine'? That way your disease agent does not have to be transmitted and arrives directly through a syringe. I saw a French virologist Luc Montagnier quoted in the alternative media as saying he had discovered that the alleged 'new' severe acute respiratory syndrome coronavirus , or SARS-CoV-2, was made artificially and included elements of the human immunodeficiency 'virus' (HIV) and a parasite that causes malaria. SARS-CoV-2 is alleged to trigger an alleged illness called Covid-19. I remembered Montagnier's name from my research years before into claims that an HIV 'retrovirus' causes AIDs – claims that were demolished by Berkeley virologist Peter Duesberg who showed that no one had ever proved that HIV causes acquired immunodeficiency syndrome or AIDS. Claims that become accepted as fact, publicly and medically, with no proof whatsoever are an ever-recurring story that profoundly applies to

'Covid'. Nevertheless, despite the lack of proof, Montagnier's team at the Pasteur Institute in Paris had a long dispute with American researcher Robert Gallo over which of them discovered and isolated the HIV 'virus' and with *no evidence* found it to cause AIDS. You will see later that there is also no evidence that any 'virus' causes any disease or that there is even such a thing as a 'virus' in the way it is said to exist. The claim to have 'isolated' the HIV 'virus' will be presented in its real context as we come to the shocking story – and it is a story – of SARS-CoV-2 and so will Montagnier's assertion that he identified the full SARS-CoV-2 genome.

Hoax in the making

We can pick up the 'Covid' story in 2010 and the publication by the Rockefeller Foundation of a document called 'Scenarios for the Future of Technology and International Development'. The inner circle of the Rockefeller family has been serving the Cult since John D. Rockefeller (1839-1937) made his fortune with Standard Oil. It is less well known that the same Rockefeller – the Bill Gates of his day – was responsible for establishing what is now referred to as 'Big Pharma', the global network of pharmaceutical companies that make outrageous profits dispensing scalpel and drug 'medicine' and are obsessed with pumping vaccines in ever-increasing number into as many human arms and backsides as possible. John D. Rockefeller was the driving force behind the creation of the 'education' system in the United States and elsewhere specifically designed to program the perceptions of generations thereafter. The Rockefeller family donated exceptionally valuable land in New York for the United Nations building and were central in establishing the World Health Organization in 1948 as an agency of the UN which was created from the start as a Trojan horse and stalking horse for world government. Now enter Bill Gates. His family and the Rockefellers have long been extremely close and I have seen genealogy which claims that if you go back far enough the two families fuse into the same bloodline. Gates has said that the Bill and Melinda Gates Foundation was inspired by the Rockefeller Foundation and why not

when both are serving the same Cult? Major tax-exempt foundations are overwhelmingly criminal enterprises in which Cult assets fund the Cult agenda in the guise of 'philanthropy' while avoiding tax in the process. Cult operatives can become mega-rich in their role of front men and women for the psychopaths at the inner core and they, too, have to be psychopaths to knowingly serve such evil. Part of the deal is that a big percentage of the wealth gleaned from representing the Cult has to be spent advancing the ambitions of the Cult and hence you have the Rockefeller Foundation, Bill and Melinda Gates Foundation (and *so* many more) and people like George Soros with his global Open Society Foundations spending their billions in pursuit of global Cult control. Gates is a global public face of the Cult with his interventions in world affairs including Big Tech influence; a central role in the 'Covid' and 'vaccine' scam; promotion of the climate change shakedown; manipulation of education; geoengineering of the skies; and his food-control agenda as the biggest owner of farmland in America, his GMO promotion and through other means. As one writer said: 'Gates monopolizes or wields disproportionate influence over the tech industry, global health and vaccines, agriculture and food policy (including biopiracy and fake food), weather modification and other climate technologies, surveillance, education and media.' The almost limitless wealth secured through Microsoft and other not-allowed-to-fail ventures (including vaccines) has been ploughed into a long, long list of Cult projects designed to enslave the entire human race. Gates and the Rockefellers have been working as one unit with the Rockefeller-established World Health Organization leading global 'Covid' policy controlled by Gates through his mouth-piece Tedros. Gates became the WHO's biggest funder when Trump announced that the American government would cease its donations, but Biden immediately said he would restore the money when he took office in January, 2021. The Gates Foundation (the Cult) owns through limitless funding the world health system and the major players across the globe in the 'Covid' hoax.

Okay, with that background we return to that Rockefeller Foundation document of 2010 headed 'Scenarios for the Future of Technology and International Development' and its 'imaginary' epidemic of a virulent and deadly influenza strain which infected 20 percent of the global population and killed eight million in seven months. The Rockefeller scenario was that the epidemic destroyed economies, closed shops, offices and other businesses and led to governments imposing fierce rules and restrictions that included mandatory wearing of face masks and body-temperature checks to enter communal spaces like railway stations and supermarkets. The document predicted that even after the height of the Rockefeller-envisaged epidemic the authoritarian rule would continue to deal with further pandemics, transnational terrorism, environmental crises and rising poverty. Now you may think that the Rockefellers are our modern-day seers or alternatively, and rather more likely, that they well knew what was planned a few years further on. Fascism had to be imposed, you see, to 'protect citizens from risk and exposure'. The Rockefeller scenario document said:

During the pandemic, national leaders around the world flexed their authority and imposed airtight rules and restrictions, from the mandatory wearing of face masks to body-temperature checks at the entries to communal spaces like train stations and supermarkets. Even after the pandemic faded, this more authoritarian control and oversight of citizens and their activities stuck and even intensified. In order to protect themselves from the spread of increasingly global problems – from pandemics and transnational terrorism to environmental crises and rising poverty – leaders around the world took a firmer grip on power.

At first, the notion of a more controlled world gained wide acceptance and approval. Citizens willingly gave up some of their sovereignty – and their privacy – to more paternalistic states in exchange for greater safety and stability. Citizens were more tolerant, and even eager, for top-down direction and oversight, and national leaders had more latitude to impose order in the ways they saw fit.

In developed countries, this heightened oversight took many forms: biometric IDs for all citizens, for example, and tighter regulation of key industries whose stability was deemed vital to national interests. In many developed countries, enforced cooperation with a suite of new regulations and agreements slowly but steadily restored both order and, importantly, economic growth.

There we have the prophetic Rockefellers in 2010 and three years later came their paper for the Global Health Summit in Beijing, China, when government representatives, the private sector, international organisations and groups met to discuss the next 100 years of 'global health'. The Rockefeller Foundation-funded paper was called 'Dreaming the Future of Health for the Next 100 Years and more prophecy ensued as it described a dystopian future: 'The abundance of data, digitally tracking and linking people may mean the 'death of privacy' and may replace physical interaction with transient, virtual connection, generating isolation and raising questions of how values are shaped in virtual networks.' Next in the 'Covid' hoax preparation sequence came a 'table top' simulation in 2018 for another 'imaginary' pandemic of a disease called Clade X which was said to kill 900 million people. The exercise was organised by the Gates-funded Johns Hopkins University's Center for Health Security in the United States and this is the very same university that has been compiling the disgustingly and systematically erroneous global figures for 'Covid' cases and deaths. Similar Johns Hopkins health crisis scenarios have included the Dark Winter exercise in 2001 and Atlantic Storm in 2005.

Nostradamus 201

For sheer predictive genius look no further prophecy-watchers than the Bill Gates-funded Event 201 held only six weeks before the 'coronavirus pandemic' is supposed to have broken out in China and Event 201 was based on a scenario of a global 'coronavirus pandemic'. Melinda Gates, the great man's missus, told the BBC that he had 'prepared for years' for a coronavirus pandemic which told us what we already knew. Nostradamugates had predicted in a TED talk in 2015 that a pandemic was coming that would kill a lot of people and demolish the world economy. My god, the man is a machine – possibly even literally. Now here he was only weeks before the real thing funding just such a simulated scenario and involving his friends and associates at Johns Hopkins, the World Economic Forum Cult-front of Klaus Schwab, the United Nations,

Johnson & Johnson, major banks, and officials from China and the Centers for Disease Control in the United States. What synchronicity – Johns Hopkins would go on to compile the fraudulent 'Covid' figures, the World Economic Forum and Schwab would push the 'Great Reset' in response to 'Covid', the Centers for Disease Control would be at the forefront of 'Covid' policy in the United States, Johnson & Johnson would produce a 'Covid vaccine', and everything would officially start just weeks later in China. Spooky, eh? They were even accurate in creating a simulation of a 'virus' pandemic because the 'real thing' would also be a simulation. Event 201 was not an exercise preparing for something that might happen; it was a rehearsal for what those in control knew was *going* to happen and very shortly. Hours of this simulation were posted on the Internet and the various themes and responses mirrored what would soon be imposed to transform human society. News stories were inserted and what they said would be commonplace a few weeks later with still more prophecy perfection. Much discussion focused on the need to deal with misinformation and the 'anti-vax movement' which is exactly what happened when the 'virus' arrived – was said to have arrived – in the West.

Cult-owned social media banned criticism and exposure of the official 'virus' narrative and when I said there *was* no 'virus' in early April, 2020, I was banned by one platform after another including YouTube, Facebook and later Twitter. The mainstream broadcast media in Britain was in effect banned from interviewing me by the Tony-Blair-created government broadcasting censor Ofcom headed by career government bureaucrat Melanie Dawes who was appointed just as the 'virus' hoax was about to play out in January, 2020. At the same time the Ickonic media platform was using Vimeo, another ultra-Zionist-owned operation, while our own player was being created and they deleted in an instant hundreds of videos, documentaries, series and shows to confirm their unbelievable vindictiveness. We had copies, of course, and they had to be restored one by one when our player was ready. These people have no class. Sabbatian Facebook promised free advertisements for the Gates-

controlled World Health Organization narrative while deleting 'false claims and conspiracy theories' to stop 'misinformation' about the alleged coronavirus. All these responses could be seen just a short while earlier in the scenarios of Event 201. Extreme censorship was absolutely crucial for the Cult because the official story was so ridiculous and unsupportable by the evidence that it could never survive open debate and the free-flow of information and opinion. If you can't win a debate then don't have one is the Cult's approach throughout history. Facebook's little boy front man – front boy – Mark Zuckerberg equated 'credible and accurate information' with official sources and exposing their lies with 'misinformation'.

Silencing those that can see

The censorship dynamic of Event 201 is now the norm with an army of narrative-supporting 'fact-checker' organisations whose entire reason for being is to tell the public that official narratives are true and those exposing them are lying. One of the most appalling of these 'fact-checkers' is called NewsGuard founded by ultra-Zionist Americans Gordon Crovitz and Steven Brill. Crovitz is a former publisher of *The Wall Street Journal*, former Executive Vice President of Dow Jones, a member of the Council on Foreign Relations (CFR), and on the board of the American Association of Rhodes Scholars. The CFR and Rhodes Scholarships, named after Rothschild agent Cecil Rhodes who plundered the gold and diamonds of South Africa for his masters and the Cult, have featured widely in my books. NewsGuard don't seem to like me for some reason – I really can't think why – and they have done all they can to have me censored and discredited which is, to quote an old British politician, like being savaged by a dead sheep. They are, however, like all in the censorship network, very well connected and funded by organisations themselves funded by, or connected to, Bill Gates. As you would expect with anything associated with Gates NewsGuard has an offshoot called HealthGuard which 'fights online health care hoaxes'. How very kind. Somehow the NewsGuard European Managing Director Anna-Sophie Harling, a remarkably young-

looking woman with no broadcasting experience and little hands-on work in journalism, has somehow secured a position on the 'Content Board' of UK government broadcast censor Ofcom. An executive of an organisation seeking to discredit dissidents of the official narratives is making decisions for the government broadcast 'regulator' about content?? Another appalling 'fact-checker' is Full Fact funded by George Soros and global censors Google and Facebook.

It's amazing how many activists in the 'fact-checking', 'anti-hate', arena turn up in government-related positions – people like UK Labour Party activist Imran Ahmed who heads the Center for Countering Digital Hate founded by people like Morgan McSweeney, now chief of staff to the Labour Party's hapless and useless 'leader' Keir Starmer. Digital Hate – which is what it really is – uses the American spelling of Center to betray its connection to a transatlantic network of similar organisations which in 2020 shapeshifted from attacking people for 'hate' to attacking them for questioning the 'Covid' hoax and the dangers of the 'Covid vaccine'. It's just a coincidence, you understand. This is one of Imran Ahmed's hysterical statements: 'I would go beyond calling anti-vaxxers conspiracy theorists to say they are an extremist group that pose a national security risk.' No one could ever accuse this prat of understatement and he's including in that those parents who are now against vaccines after their children were damaged for life or killed by them. He's such a nice man. Ahmed does the rounds of the Woke media getting soft-ball questions from spineless 'journalists' who never ask what right he has to campaign to destroy the freedom of speech of others while he demands it for himself. There also seems to be an overrepresentation in Ofcom of people connected to the narrative-worshipping BBC. This incredible global network of narrative-support was super-vital when the 'Covid' hoax was played in the light of the mega-whopper lies that have to be defended from the spotlight cast by the most basic intelligence.

Setting the scene

The Cult plays the long game and proceeds step-by-step ensuring that everything is in place before major cards are played and they don't come any bigger than the 'Covid' hoax. The psychopaths can't handle events where the outcome isn't certain and as little as possible – preferably nothing – is left to chance. Politicians, government and medical officials who would follow direction were brought to illusory power in advance by the Cult web whether on the national stage or others like state governors and mayors of America. For decades the dynamic between officialdom, law enforcement and the public was changed from one of service to one of control and dictatorship. Behaviour manipulation networks established within government were waiting to impose the coming 'Covid' rules and regulations specifically designed to subdue and rewire the psyche of the people in the guise of protecting health. These included in the UK the Behavioural Insights Team part-owned by the British government Cabinet Office; the Scientific Pandemic Insights Group on Behaviours (SPI-B); and a whole web of intelligence and military groups seeking to direct the conversation on social media and control the narrative. Among them are the cyberwarfare (on the people) 77th Brigade of the British military which is also coordinated through the Cabinet Office as civilian and military leadership continues to combine in what they call the Fusion Doctrine. The 77th Brigade is a British equivalent of the infamous Israeli (Sabbatian) military cyberwarfare and Internet manipulation operation Unit 8200 which I expose at length in *The Trigger*. Also carefully in place were the medical and science advisers to government – many on the payroll past or present of Bill Gates – and a whole alternative structure of unelected government stood by to take control when elected parliaments were effectively closed down once the 'Covid' card was slammed on the table. The structure I have described here and so much more was installed in every major country through the Cult networks. The top-down control hierarchy looks like this: The Cult – Cult-owned Gates – the World Health Organization and Tedros – Gates-funded or controlled chief medical officers and science 'advisers' (dictators) in each country –

political 'leaders'– law enforcement – The People. Through this simple global communication and enforcement structure the policy of the Cult could be imposed on virtually the entire human population so long as they acquiesced to the fascism. With everything in place it was time for the button to be pressed in late 2019/early 2020.

These were the prime goals the Cult had to secure for its will to prevail:

1) Locking down economies, closing all but designated 'essential' businesses (Cult-owned corporations were 'essential'), and putting the population under house arrest was an imperative to destroy independent income and employment and ensure dependency on the Cult-controlled state in the Hunger Games Society. Lockdowns had to be established as the global blueprint from the start to respond to the 'virus' and followed by pretty much the entire world.

2) The global population had to be terrified into believing in a deadly 'virus' that didn't actually exist so they would unquestioningly obey authority in the belief that authority must know how best to protect them and their families. Software salesman Gates would suddenly morph into the world's health expert and be promoted as such by the Cult-owned media.

3) A method of testing that wasn't testing for the 'virus', but was only claimed to be, had to be in place to provide the illusion of 'cases' and subsequent 'deaths' that had a very different cause to the 'Covid-19' that would be scribbled on the death certificate.

4) Because there was no 'virus' and the great majority testing positive with a test not testing for the 'virus' would have no symptoms of anything the lie had to be sold that people without symptoms (without the 'virus') could still pass it on to others. This was crucial to justify for the first time quarantining – house arresting – healthy people. Without this the economy-destroying lockdown of *everybody* could not have been credibly sold.

5) The 'saviour' had to be seen as a vaccine which beyond evil drug companies were working like angels of mercy to develop as quickly as possible, with all corners cut, to save the day. The public must absolutely not know that the 'vaccine' had nothing to do with a 'virus' or that the contents were ready and waiting with a very different motive long before the 'Covid' card was even lifted from the pack.

I said in March, 2020, that the 'vaccine' would have been created way ahead of the 'Covid' hoax which justified its use and the following December an article in the New York *Intelligencer* magazine said the Moderna 'vaccine' had been 'designed' by

January, 2020. This was 'before China had even acknowledged that the disease could be transmitted from human to human, more than a week before the first confirmed coronavirus case in the United States'. The article said that by the time the first American death was announced a month later 'the vaccine had already been manufactured and shipped to the National Institutes of Health for the beginning of its Phase I clinical trial'. The 'vaccine' was actually 'designed' long before that although even with this timescale you would expect the article to ask how on earth it could have been done that quickly. Instead it asked why the 'vaccine' had not been rolled out then and not months later. Journalism in the mainstream is truly dead. I am going to detail in the next chapter why the 'virus' has never existed and how a hoax on that scale was possible, but first the foundation on which the Big Lie of 'Covid' was built.

The test that doesn't test

Fraudulent 'testing' is the bottom line of the whole 'Covid' hoax and was the means by which a 'virus' that did not exist *appeared* to exist. They could only achieve this magic trick by using a test not testing for the 'virus'. To use a test that *was* testing for the 'virus' would mean that every test would come back negative given there was no 'virus'. They chose to exploit something called the RT-PCR test invented by American biochemist Kary Mullis in the 1980s who said publicly that his PCR test … *cannot detect infectious disease*. Yes, the 'test' used worldwide to detect infectious 'Covid' to produce all the illusory 'cases' and 'deaths' compiled by Johns Hopkins and others *cannot detect infectious disease*. This fact came from the mouth of the man who invented PCR and was awarded the Nobel Prize in Chemistry in 1993 for doing so. Sadly, and incredibly conveniently for the Cult, Mullis died in August, 2019, at the age of 74 just before his test would be fraudulently used to unleash fascism on the world. He was said to have died from pneumonia which was an irony in itself. A few months later he would have had 'Covid-19' on his death certificate. I say the timing of his death was convenient because had he lived Mullis, a brilliant, honest and decent man, would have been

vociferously speaking out against the use of his test to detect 'Covid' when it was never designed, or able, to do that. I know that to be true given that Mullis made the same point when his test was used to 'detect' – not detect – HIV. He had been seriously critical of the Gallo/Montagnier claim to have isolated the HIV 'virus' and shown it to cause AIDS for which Mullis said there was no evidence. AIDS is actually not a disease but a series of diseases from which people die all the time. When they die from those *same diseases* after a positive 'test' for HIV then AIDS goes on their death certificate. I think I've heard that before somewhere. Countries instigated a policy with 'Covid' that anyone who tested positive with a test not testing for the 'virus' and died of any other cause within 28 days and even longer 'Covid-19' had to go on the death certificate. Cases have come from the test that can't test for infectious disease and the deaths are those who have died of *anything* after testing positive with a test not testing for the 'virus'. I'll have much more later about the death certificate scandal.

Mullis was deeply dismissive of the now US 'Covid' star Anthony Fauci who he said was a liar who didn't know anything about anything – 'and I would say that to his face – nothing.' He said of Fauci: 'The man thinks he can take a blood sample, put it in an electron microscope and if it's got a virus in there you'll know it – he doesn't understand electron microscopy and he doesn't understand medicine and shouldn't be in a position like he's in.' That position, terrifyingly, has made him the decider of 'Covid' fascism policy on behalf of the Cult in his role as director since 1984 of the National Institute of Allergy and Infectious Diseases (NIAID) while his record of being wrong is laughable; but being wrong, so long as it's the *right kind* of wrong, is why the Cult loves him. He'll say anything the Cult tells him to say. Fauci was made Chief Medical Adviser to the President immediately Biden took office. Biden was installed in the White House by Cult manipulation and one of his first decisions was to elevate Fauci to a position of even more control. This is a coincidence? Yes, and I identify as a flamenco dancer called Lola. How does such an incompetent criminal like Fauci remain in that

pivotal position in American health since *the 1980s*? When you serve the Cult it looks after you until you are surplus to requirements. Kary Mullis said prophetically of Fauci and his like: 'Those guys have an agenda and it's not an agenda we would like them to have … they make their own rules, they change them when they want to, and Tony Fauci does not mind going on television in front of the people who pay his salary and lie directly into the camera.' Fauci has done that almost daily since the 'Covid' hoax began. Lying is in Fauci's DNA. To make the situation crystal clear about the PCR test this is a direct quote from its inventor Kary Mullis:

> It [the PCR test] doesn't tell you that you're sick and doesn't tell you that the thing you ended up with was really going to hurt you ...'

Ask yourself why governments and medical systems the world over have been using this very test to decide who is 'infected' with the SARS-CoV-2 'virus' and the alleged disease it allegedly causes, 'Covid-19'. The answer to that question will tell you what has been going on. By the way, here's a little show-stopper – the 'new' SARS-CoV-2 'virus' was 'identified' as such right from the start using … *the PCR test not testing for the 'virus'*. If you are new to this and find that shocking then stick around. I have hardly started yet. Even worse, other 'tests', like the 'Lateral Flow Device' (LFD), are considered so useless that they have to be *confirmed* by the PCR test! Leaked emails written by Ben Dyson, adviser to UK 'Health' Secretary Matt Hancock, said they were 'dangerously unreliable'. Dyson, executive director of strategy at the Department of Health, wrote: 'As of today, someone who gets a positive LFD result in (say) London has at best a 25 per cent chance of it being a true positive, but if it is a self-reported test potentially as low as 10 per cent (on an optimistic assumption about specificity) or as low as 2 per cent (on a more pessimistic assumption).' These are the 'tests' that schoolchildren and the public are being urged to have twice a week or more and have to isolate if they get a positive. Each fake positive goes in the statistics as a 'case' no matter how ludicrously inaccurate and the

'cases' drive lockdown, masks and the pressure to 'vaccinate'. The government said in response to the email leak that the 'tests' were accurate which confirmed yet again what shocking bloody liars they are. The real false positive rate is *100 percent* as we'll see. In another 'you couldn't make it up' the UK government agreed to pay £2.8 billion to California's Innova Medical Group to supply the irrelevant lateral flow tests. The company's primary test-making centre is in China. Innova Medical Group, established in March, 2020, is owned by Pasaca Capital Inc, chaired by Chinese-American millionaire Charles Huang who was born in Wuhan.

How it works – and how it doesn't

The RT-PCR test, known by its full title of Polymerase chain reaction, is used across the world to make millions, even billions, of copies of a DNA/RNA genetic information sample. The process is called 'amplification' and means that a tiny sample of genetic material is amplified to bring out the detailed content. I stress that it is not testing for an infectious disease. It is simply amplifying a sample of genetic material. In the words of Kary Mullis: 'PCR is … just a process that's used to make a whole lot of something out of something.' To emphasise the point companies that make the PCR tests circulated around the world to 'test' for 'Covid' warn on the box that it can't be used to detect 'Covid' or infectious disease and is for research purposes only. It's okay, rest for a minute and you'll be fine. This is the test that produces the 'cases' and 'deaths' that have been used to destroy human society. All those global and national medical and scientific 'experts' demanding this destruction to 'save us' *KNOW* that the test is not testing for the 'virus' and the cases and deaths they claim to be real are an almost unimaginable fraud. Every one of them and so many others including politicians and psychopaths like Gates and Tedros must be brought before Nuremburg-type trials and jailed for the rest of their lives. The more the genetic sample is amplified by PCR the more elements of that material become sensitive to the test and by that I don't mean sensitive for a 'virus' but for elements of the genetic material which

is *naturally* in the body or relates to remnants of old conditions of various kinds lying dormant and causing no disease. Once the amplification of the PCR reaches a certain level *everyone* will test positive. So much of the material has been made sensitive to the test that everyone will have some part of it in their body. Even lying criminals like Fauci have said that once PCR amplifications pass 35 cycles everything will be a false positive that cannot be trusted for the reasons I have described. I say, like many proper doctors and scientists, that 100 percent of the 'positives' are false, but let's just go with Fauci for a moment.

He says that any amplification over 35 cycles will produce false positives and yet the US Centers for Disease Control (CDC) and Food and Drug Administration (FDA) have recommended up to *40 cycles* and the National Health Service (NHS) in Britain admitted in an internal document for staff that it was using *45 cycles* of amplification. A long list of other countries has been doing the same and at least one 'testing' laboratory has been using *50 cycles*. Have you ever heard a doctor, medical 'expert' or the media ask what level of amplification has been used to claim a 'positive'. The 'test' comes back 'positive' and so you have the 'virus', end of story. Now we can see how the government in Tanzania could send off samples from a goat and a pawpaw fruit under human names and both came back positive for 'Covid-19'. Tanzania president John Magufuli mocked the 'Covid' hysteria, the PCR test and masks and refused to import the DNA-manipulating 'vaccine'. The Cult hated him and an article sponsored by the Bill Gates Foundation appeared in the London *Guardian* in February, 2021, headed 'It's time for Africa to rein in Tanzania's anti-vaxxer president'. Well, 'reined in' he shortly was. Magufuli appeared in good health, but then, in March, 2021, he was dead at 61 from 'heart failure'. He was replaced by Samia Hassan Suhulu who is connected to Klaus Schwab's World Economic Forum and she immediately reversed Magufuli's 'Covid' policy. A sample of cola tested positive for 'Covid' with the PCR test in Germany while American actress and singer-songwriter Erykah Badu tested positive in one nostril and negative in the other. Footballer Ronaldo called

the PCR test 'bullshit' after testing positive three times and being forced to quarantine and miss matches when there was nothing wrong with him. The mantra from Tedros at the World Health Organization and national governments (same thing) has been test, test, test. They know that the more tests they can generate the more fake 'cases' they have which go on to become 'deaths' in ways I am coming to. The UK government has its Operation Moonshot planned to test multiple millions every day in workplaces and schools with free tests for everyone to use twice a week at home in line with the Cult plan from the start to make testing part of life. A government advertisement for an 'Interim Head of Asymptomatic Testing Communication' said the job included responsibility for delivering a 'communications strategy' (propaganda) 'to support the expansion of asymptomatic testing that *'normalises testing as part of everyday life'*. More tests means more fake 'cases', 'deaths' and fascism. I have heard of, and from, many people who booked a test, couldn't turn up, and yet got a positive result through the post for a test they'd never even had. The whole thing is crazy, but for the Cult there's method in the madness. Controlling and manipulating the level of amplification of the test means the authorities can control whenever they want the number of apparent 'cases' and 'deaths'. If they want to justify more fascist lockdown and destruction of livelihoods they keep the amplification high. If they want to give the illusion that lockdowns and the 'vaccine' are working then they lower the amplification and 'cases' and 'deaths' will appear to fall. In January, 2021, the Cult-owned World Health Organization suddenly warned laboratories about over-amplification of the test and to lower the threshold. Suddenly headlines began appearing such as: 'Why ARE "Covid" cases plummeting?' This was just when the vaccine rollout was underway and I had predicted months before they would make cases appear to fall through amplification tampering when the 'vaccine' came. These people are so predictable.

Cow vaccines?

The question must be asked of what is on the test swabs being poked far up the nose of the population to the base of the brain? A nasal swab punctured one woman's brain and caused it to leak fluid. Most of these procedures are being done by people with little training or medical knowledge. Dr Lorraine Day, former orthopaedic trauma surgeon and Chief of Orthopaedic Surgery at San Francisco General Hospital, says the tests are really a *'vaccine'*. Cows have long been vaccinated this way. She points out that masks have to cover the nose and the mouth where it is claimed the 'virus' exists in saliva. Why then don't they take saliva from the mouth as they do with a DNA test instead of pushing a long swab up the nose towards the brain? The ethmoid bone separates the nasal cavity from the brain and within that bone is the cribriform plate. Dr Day says that when the swab is pushed up against this plate and twisted the procedure is 'depositing things back there'. She claims that among these 'things' are nanoparticles that can enter the brain. Researchers have noted that a team at the Gates-funded Johns Hopkins have designed tiny, star-shaped micro-devices that can latch onto intestinal mucosa and release drugs into the body. Mucosa is the thin skin that covers the inside surface of parts of the body such as *the nose* and mouth and produces mucus to protect them. The Johns Hopkins micro-devices are called 'theragrippers' and were 'inspired' by a parasitic worm that digs its sharp teeth into a host's intestines. Nasal swabs are also coated in the sterilisation agent ethylene oxide. The US National Cancer Institute posts this explanation on its website:

> At room temperature, ethylene oxide is a flammable colorless gas with a sweet odor. It is used primarily to produce other chemicals, including antifreeze. In smaller amounts, ethylene oxide is used as a pesticide and a sterilizing agent. The ability of ethylene oxide to damage DNA makes it an effective sterilizing agent but also accounts for its cancer-causing activity.

The Institute mentions lymphoma and leukaemia as cancers most frequently reported to be associated with occupational exposure to ethylene oxide along with stomach and breast cancers. How does anyone think this is going to work out with the constant testing

regime being inflicted on adults and children at home and at school that will accumulate in the body anything that's on the swab?

Doctors know best

It is vital for people to realise that 'hero' doctors 'know' only what the Big Pharma-dominated medical authorities tell them to 'know' and if they refuse to 'know' what they are told to 'know' they are out the door. They are mostly not physicians or healers, but repeaters of the official narrative – or else. I have seen alleged professional doctors on British television make shocking statements that we are supposed to take seriously. One called 'Dr' Amir Khan, who is actually telling patients how to respond to illness, said that men could take the birth pill to 'help slow down the effects of Covid-19'. In March, 2021, another ridiculous 'Covid study' by an American doctor proposed injecting men with the female sex hormone progesterone as a 'Covid' treatment. British doctor Nighat Arif told the BBC that face coverings were now going to be part of ongoing normal. Yes, the vaccine protects you, she said (evidence?) … but the way to deal with viruses in the community was always going to come down to hand washing, face covering and keeping a physical distance. That's not what we were told before the 'vaccine' was circulating. Arif said she couldn't imagine ever again going on the underground or in a lift without a mask. I was just thanking my good luck that she was not my doctor when she said – in March, 2021 – that if 'we are *behaving* and we are doing all the right things' she thought we could 'have our nearest and dearest around us at home … around *Christmas* and N*ew Year!* Her patronising delivery was the usual school teacher talking to six-year-olds as she repeated every government talking point and probably believed them all. If we have learned anything from the 'Covid' experience surely it must be that humanity's perception of doctors needs a fundamental rethink. NHS 'doctor' Sara Kayat told her television audience that the 'Covid vaccine' would '100 percent prevent hospitalisation and death'. Not even Big Pharma claimed that. We have to stop taking 'experts' at their word without question when so many of them are

clueless and only repeating the party line on which their careers depend. That is not to say there are not brilliants doctors – there are and I have spoken to many of them since all this began – but you won't see them in the mainstream media or quoted by the psychopaths and yes-people in government.

Remember the name – Christian Drosten

German virologist Christian Drosten, Director of Charité Institute of Virology in Berlin, became a national star after the pandemic hoax began. He was feted on television and advised the German government on 'Covid' policy. Most importantly to the wider world Drosten led a group that produced the 'Covid' testing protocol for the PCR test. What a remarkable feat given the PCR cannot test for infectious disease and even more so when you think that Drosten said that his method of testing for SARS-CoV-2 was developed 'without having virus material available'. *He developed a test for a 'virus' that he didn't have and had never seen*. Let that sink in as you survey the global devastation that came from what he did. The whole catastrophe of Drosten's 'test' was based on the alleged genetic sequence published by Chinese scientists on the Internet. We will see in the next chapter that this alleged 'genetic sequence' has never been produced by China or anyone and cannot be when there *is no* SARS-CoV-2. Drosten, however, doesn't seem to let little details like that get in the way. He was the lead author with Victor Corman from the same Charité Hospital of the paper 'Detection of 2019 novel coronavirus (2019-nCoV) by real-time PCR' published in a magazine called *Eurosurveillance*. This became known as the Corman-Drosten paper. In November, 2020, with human society devastated by the effects of the Corman-Drosten test baloney, the protocol was publicly challenged by 22 international scientists and independent researchers from Europe, the United States, and Japan. Among them were senior molecular geneticists, biochemists, immunologists, and microbiologists. They produced a document headed 'External peer review of the RTPCR test to detect SARS-Cov-2 Reveals 10 Major Flaws At The Molecular and Methodological Level: Consequences

For False-Positive Results'. The flaws in the Corman-Drosten test included the following:

- The test is non-specific because of erroneous design
- Results are enormously variable
- The test is unable to discriminate between the whole 'virus' and viral fragments
- It doesn't have positive or negative controls
- The test lacks a standard operating procedure
- It is unsupported by proper peer view

The scientists said the PCR 'Covid' testing protocol was not founded on science and they demanded the Corman-Drosten paper be retracted by *Eurosurveillance*. They said all present and previous Covid deaths, cases, and 'infection rates' should be subject to a massive retroactive inquiry. Lockdowns and travel restrictions should be reviewed and relaxed and those diagnosed through PCR to have 'Covid-19' should not be forced to isolate. Dr Kevin Corbett, a health researcher and nurse educator with a long academic career producing a stream of peer-reviewed publications at many UK universities, made the same point about the PCR test debacle. He said of the scientists' conclusions: 'Every scientific rationale for the development of that test has been totally destroyed by this paper. It's like Hiroshima/Nagasaki to the Covid test.' He said that China hadn't given them an isolated 'virus' when Drosten developed the test. Instead they had developed the test from *a sequence in a gene bank*.' Put another way … *they made it up*! The scientists were supported in this contention by a Portuguese appeals court which ruled in November, 2020, that PCR tests are unreliable and it is unlawful to quarantine people based solely on a PCR test. The point about China not providing an isolated virus must be true when the 'virus' has never been isolated to this day and the consequences of that will become clear. Drosten and company produced this useless 'protocol' right on cue in January, 2020, just as the 'virus' was said to

be moving westward and it somehow managed to successfully pass a peer-review in 24 hours. In other words there was no peer-review for a test that would be used to decide who had 'Covid' and who didn't across the world. The Cult-created, Gates-controlled World Health Organization immediately recommended all its nearly 200 member countries to use the Drosten PCR protocol to detect 'cases' and 'deaths'. The sting was underway and it continues to this day.

So who is this Christian Drosten that produced the means through which death, destruction and economic catastrophe would be justified? His education background, including his doctoral thesis, would appear to be somewhat shrouded in mystery and his track record is dire as with another essential player in the 'Covid' hoax, the Gates-funded Professor Neil Ferguson at the Gates-funded Imperial College in London of whom more shortly. Drosten predicted in 2003 that the alleged original SARS 'virus' (SARS-1') was an epidemic that could have serious effects on economies and an effective vaccine would take at least two years to produce. Drosten's answer to every alleged 'outbreak' is a vaccine which you won't be shocked to know. What followed were just 774 official deaths worldwide and none in Germany where there were only nine cases. That is even if you believe there ever was a SARS 'virus' when the evidence is zilch and I will expand on this in the next chapter. Drosten claims to be co-discoverer of 'SARS-1' and developed a test for it in 2003. He was screaming warnings about 'swine flu' in 2009 and how it was a widespread infection far more severe than any dangers from a vaccine could be and people should get vaccinated. It would be helpful for Drosten's vocal chords if he simply recorded the words 'the virus is deadly and you need to get vaccinated' and copies could be handed out whenever the latest made-up threat comes along. Drosten's swine flu epidemic never happened, but Big Pharma didn't mind with governments spending hundreds of millions on vaccines that hardly anyone bothered to use and many who did wished they hadn't. A study in 2010 revealed that the risk of dying from swine flu, or H1N1, was no higher than that of the annual seasonal flu which is what at least most of 'it' really was as in

the case of 'Covid-19'. A media investigation into Drosten asked how with such a record of inaccuracy he could be *the* government adviser on these issues. The answer to that question is the same with Drosten, Ferguson and Fauci – they keep on giving the authorities the 'conclusions' and 'advice' they want to hear. Drosten certainly produced the goods for them in January, 2020, with his PCR protocol garbage and provided the foundation of what German internal medicine specialist Dr Claus Köhnlein, co-author of *Virus Mania*, called the 'test pandemic'. The 22 scientists in the *Eurosurveillance* challenge called out conflicts of interest within the Drosten 'protocol' group and with good reason. Olfert Landt, a regular co-author of Drosten 'studies', owns the biotech company TIB Molbiol Syntheselabor GmbH in Berlin which manufactures and sells the tests that Drosten and his mates come up with. They have done this with SARS, Enterotoxigenic E. coli (ETEC), MERS, Zika 'virus', yellow fever, and now 'Covid'. Landt told the *Berliner Zeitung* newspaper:

> The testing, design and development came from the Charité [Drosten and Corman]. We simply implemented it immediately in the form of a kit. And if we don't have the virus, which originally only existed in Wuhan, we can make a synthetic gene to simulate the genome of the virus. That's what we did very quickly.

This is more confirmation that the Drosten test was designed without access to the 'virus' and only a synthetic simulation which is what SARS-CoV-2 really is – a computer-generated synthetic fiction. It's quite an enterprise they have going here. A Drosten team decides what the test for something should be and Landt's biotech company flogs it to governments and medical systems across the world. His company must have made an absolute fortune since the 'Covid' hoax began. Dr Reiner Fuellmich, a prominent German consumer protection trial lawyer in Germany and California, is on Drosten's case and that of Tedros at the World Health Organization for crimes against humanity with a class-action lawsuit being prepared in the United States and other legal action in Germany.

Why China?

Scamming the world with a 'virus' that doesn't exist would seem impossible on the face of it, but not if you have control of the relatively few people that make policy decisions and the great majority of the global media. Remember it's not about changing 'real' reality it's about controlling *perception* of reality. You don't have to make something happen you only have make people *believe* that it's happening. Renegade Minds understand this and are therefore much harder to swindle. 'Covid-19' is not a 'real' 'virus'. It's a mind virus, like a computer virus, which has infected the minds, not the bodies, of billions. It all started, publically at least, in China and that alone is of central significance. The Cult was behind the revolution led by its asset Mao Zedong, or Chairman Mao, which established the People's Republic of China on October 1st, 1949. It should have been called The Cult's Republic of China, but the name had to reflect the recurring illusion that vicious dictatorships are run by and for the people (see all the 'Democratic Republics' controlled by tyrants). In the same way we have the 'Biden' Democratic Republic of America officially ruled by a puppet tyrant (at least temporarily) on behalf of Cult tyrants. The creation of Mao's merciless communist/fascist dictatorship was part of a frenzy of activity by the Cult at the conclusion of World War Two which, like the First World War, it had instigated through its assets in Germany, Britain, France, the United States and elsewhere. Israel was formed in 1948; the Soviet Union expanded its 'Iron Curtain' control, influence and military power with the Warsaw Pact communist alliance in 1955; the United Nations was formed in 1945 as a Cult precursor to world government; and a long list of world bodies would be established including the World Health Organization (1948), World Trade Organization (1948 under another name until 1995), International Monetary Fund (1945) and World Bank (1944). Human society was redrawn and hugely centralised in the global Problem-Reaction-Solution that was World War Two. All these changes were significant. Israel would become the headquarters of the Sabbatians

and the revolution in China would prepare the ground and control system for the events of 2019/2020.

Renegade Minds know there are no borders except for public consumption. The Cult is a seamless, borderless global entity and to understand the game we need to put aside labels like borders, nations, countries, communism, fascism and democracy. These delude the population into believing that countries are ruled within their borders by a government of whatever shade when these are mere agencies of a global power. America's illusion of democracy and China's communism/fascism are subsidiaries – vehicles – for the same agenda. We may hear about conflict and competition between America and China and on the lower levels that will be true; but at the Cult level they are branches of the same company in the way of the McDonald's example I gave earlier. I have tracked in the books over the years support by US governments of both parties for Chinese Communist Party infiltration of American society through allowing the sale of land, even military facilities, and the acquisition of American business and university influence. All this is underpinned by the infamous stealing of intellectual property and technological know-how. Cult-owned Silicon Valley corporations waive their fraudulent 'morality' to do business with human-rights-free China; Cult-controlled Disney has become China's PR department; and China in effect owns 'American' sports such as basketball which depends for much of its income on Chinese audiences. As a result any sports player, coach or official speaking out against China's horrific human rights record is immediately condemned or fired by the China-worshipping National Basketball Association. One of the first acts of China-controlled Biden was to issue an executive order telling federal agencies to stop making references to the 'virus' by the 'geographic location of its origin'. Long-time Congressman Jerry Nadler warned that criticising China, America's biggest rival, leads to hate crimes against Asian people in the United States. So shut up you bigot. China is fast closing in on Israel as a country that must not be criticised which is apt, really, given that Sabbatians control them both. The two countries have

developed close economic, military, technological and strategic ties which include involvement in China's 'Silk Road' transport and economic initiative to connect China with Europe. Israel was the first country in the Middle East to recognise the establishment of Mao's tyranny in 1950 months after it was established.

Project Wuhan – the 'Covid' Psyop

I emphasise again that the Cult plays the long game and what is happening to the world today is the result of centuries of calculated manipulation following a script to take control step-by-step of every aspect of human society. I will discuss later the common force behind all this that has spanned those centuries and thousands of years if the truth be told. Instigating the Mao revolution in China in 1949 with a 2020 'pandemic' in mind is not only how they work – the 71 years between them is really quite short by the Cult's standards of manipulation preparation. The reason for the Cult's Chinese revolution was to create a fiercely-controlled environment within which an extreme structure for human control could be incubated to eventually be unleashed across the world. We have seen this happen since the 'pandemic' emerged from China with the Chinese control-structure founded on AI technology and tyrannical enforcement sweep across the West. Until the moment when the Cult went for broke in the West and put its fascism on public display Western governments had to pay some lip-service to freedom and democracy to not alert too many people to the tyranny-in-the-making. Freedoms were more subtly eroded and power centralised with covert government structures put in place waiting for the arrival of 2020 when that smokescreen of 'freedom' could be dispensed with. The West was not able to move towards tyranny before 2020 anything like as fast as China which was created as a tyranny and had no limits on how fast it could construct the Cult's blueprint for global control. When the time came to impose that structure on the world it was the same Cult-owned Chinese communist/fascist government that provided the excuse – the 'Covid pandemic'. It was absolutely crucial to the Cult plan for the Chinese response to the 'pandemic' –

draconian lockdowns of the entire population – to become the blueprint that Western countries would follow to destroy the livelihoods and freedom of their people. This is why the Cult-owned, Gates-owned, WHO Director-General Tedros said early on:

> The Chinese government is to be congratulated for the extraordinary measures it has taken to contain the outbreak. China is actually setting a new standard for outbreak response and it is not an exaggeration.

Forbes magazine said of China: '… those measures protected untold millions from getting the disease'. The Rockefeller Foundation 'epidemic scenario' document in 2010 said 'prophetically':

> However, a few countries did fare better – China in particular. The Chinese government's quick imposition and enforcement of mandatory quarantine for all citizens, as well as its instant and near-hermetic sealing off of all borders, saved millions of lives, stopping the spread of the virus far earlier than in other countries and enabling a swifter post-pandemic recovery.

Once again – *spooky*.

The first official story was the 'bat theory' or rather the bat diversion. The source of the 'virus outbreak' we were told was a ''wet market' in Wuhan where bats and other animals are bought and eaten in horrifically unhygienic conditions. Then another story emerged through the alternative media that the 'virus' had been released on purpose or by accident from a BSL-4 (biosafety level 4) laboratory in Wuhan not far from the wet market. The lab was reported to create and work with lethal concoctions and bioweapons. Biosafety level 4 is the highest in the World Health Organization system of safety and containment. Renegade Minds are aware of what I call designer manipulation. The ideal for the Cult is for people to buy its prime narrative which in the opening salvoes of the 'pandemic' was the wet market story. It knows, however, that there is now a considerable worldwide alternative media of researchers sceptical of anything governments say and they are often given a version of events in a form they can perceive as credible while misdirecting them from the real truth. In this case let them

think that the conspiracy involved is a 'bioweapon virus' released from the Wuhan lab to keep them from the real conspiracy – *there is no 'virus'*. The WHO's current position on the source of the outbreak at the time of writing appears to be: 'We haven't got a clue, mate.' This is a good position to maintain mystery and bewilderment. The inner circle will know where the 'virus' came from – *nowhere*. The bottom line was to ensure the public believed there *was* a 'virus' and it didn't much matter if they thought it was natural or had been released from a lab. The belief that there was a 'deadly virus' was all that was needed to trigger global panic and fear. The population was terrified into handing their power to authority and doing what they were told. They had to or they were 'all gonna die'.

In March, 2020, information began to come my way from real doctors and scientists and my own additional research which had my intuition screaming: 'Yes, that's it! *There is no virus*.' The 'bioweapon' was not the 'virus'; it was the *'vaccine'* already being talked about that would be the bioweapon. My conclusion was further enhanced by happenings in Wuhan. The 'virus' was said to be sweeping the city and news footage circulated of people collapsing in the street (which they've never done in the West with the same 'virus'). The Chinese government was building 'new hospitals' in a matter of ten days to 'cope with demand' such was the virulent nature of the 'virus'. Yet in what seemed like no time the 'new hospitals' closed – even if they even opened – and China declared itself 'virus-free'. It was back to business as usual. This was more propaganda to promote the Chinese draconian lockdowns in the West as the way to 'beat the virus'. Trouble was that we subsequently had lockdown after lockdown, but never business as usual. As the people of the West and most of the rest of the world were caught in an ever-worsening spiral of lockdown, social distancing, masks, isolated old people, families forced apart, and livelihood destruction, it was party-time in Wuhan. Pictures emerged of thousands of people enjoying pool parties and concerts. It made no sense until you realised there never was a 'virus' and the

whole thing was a Cult set-up to transform human society out of one its major global strongholds – China.

How is it possible to deceive virtually the entire world population into believing there is a deadly virus when there is not even a 'virus' let alone a deadly one? It's nothing like as difficult as you would think and that's clearly true because it happened.

Postscript: See end of book Postscript for more on the 'Wuhan lab virus release' story which the authorities and media were pushing heavily in the summer of 2021 to divert attention from the truth that the 'Covid virus' is pure invention.

CHAPTER FIVE

There *is no* 'virus'

You can fool some of the people all of the time, and all of the people some of the time, but you cannot fool all of the people all of the time
Abraham Lincoln

The greatest form of mind control is repetition. The more you repeat the same mantra of alleged 'facts' the more will accept them to be true. It becomes an 'everyone knows that, mate'. If you can also censor any other version or alternative to your alleged 'facts' you are pretty much home and cooking.

By the start of 2020 the Cult owned the global mainstream media almost in its entirety to spew out its 'Covid' propaganda and ignore or discredit any other information and view. Cult-owned social media platforms in Cult-owned Silicon Valley were poised and ready to unleash a campaign of ferocious censorship to obliterate all but the official narrative. To complete the circle many demands for censorship by Silicon Valley were led by the mainstream media as 'journalists' became full-out enforcers for the Cult both as propagandists and censors. Part of this has been the influx of young people straight out of university who have become 'journalists' in significant positions. They have no experience and a headful of programmed perceptions from their years at school and university at a time when today's young are the most perceptually-targeted generations in known human history given the insidious impact of technology. They enter the media perceptually prepared and ready to repeat the narratives of the system that programmed them to

repeat its narratives. The BBC has a truly pathetic 'specialist disinformation reporter' called Marianna Spring who fits this bill perfectly. She is clueless about the world, how it works and what is really going on. Her role is to discredit anyone doing the job that a proper journalist would do and system-serving hacks like Spring wouldn't dare to do or even see the need to do. They are too busy licking the arse of authority which can never be wrong and, in the case of the BBC propaganda programme, *Panorama*, contacting payments systems such as PayPal to have a donations page taken down for a film company making documentaries questioning vaccines. Even the BBC soap opera *EastEnders* included a disgracefully biased scene in which an inarticulate white working class woman was made to look foolish for questioning the 'vaccine' while a well-spoken black man and Asian woman promoted the government narrative. It ticked every BBC box and the fact that the black and minority community was resisting the 'vaccine' had nothing to do with the way the scene was written. The BBC has become a disgusting tyrannical propaganda and censorship operation that should be defunded and disbanded and a free media take its place with a brief to stop censorship instead of demanding it. A BBC 'interview' with Gates goes something like: 'Mr Gates, sir, if I can call you sir, would you like to tell our audience why you are such a great man, a wonderful humanitarian philanthropist, and why you should absolutely be allowed as a software salesman to decide health policy for approaching eight billion people? Thank you, sir, please sir.' Propaganda programming has been incessant and merciless and when all you hear is the same story from the media, repeated by those around you who have only heard the same story, is it any wonder that people on a grand scale believe absolute mendacious garbage to be true? You are about to see, too, why this level of information control is necessary when the official 'Covid' narrative is so nonsensical and unsupportable by the evidence.

Structure of Deceit

The pyramid structure through which the 'Covid' hoax has been manifested is very simple and has to be to work. As few people as possible have to be involved with full knowledge of what they are doing – and why – or the real story would get out. At the top of the pyramid are the inner core of the Cult which controls Bill Gates who, in turn, controls the World Health Organization through his pivotal funding and his puppet Director-General mouthpiece, Tedros. Before he was appointed Tedros was chair of the Gates-founded Global Fund to 'fight against AIDS, tuberculosis and malaria', a board member of the Gates-funded 'vaccine alliance' GAVI, and on the board of another Gates-funded organisation. Gates owns him and picked him for a specific reason – Tedros is a crook and worse. 'Dr' Tedros (he's not a medical doctor, the first WHO chief not to be) was a member of the tyrannical Marxist government of Ethiopia for decades with all its human rights abuses. He has faced allegations of corruption and misappropriation of funds and was exposed three times for covering up cholera epidemics while Ethiopia's health minister. Tedros appointed the mass-murdering genocidal Zimbabwe dictator Robert Mugabe as a WHO goodwill ambassador for public health which, as with Tedros, is like appointing a psychopath to run a peace and love campaign. The move was so ridiculous that he had to drop Mugabe in the face of widespread condemnation. American economist David Steinman, a Nobel peace prize nominee, lodged a complaint with the International Criminal Court in The Hague over alleged genocide by Tedros when he was Ethiopia's foreign minister. Steinman says Tedros was a 'crucial decision maker' who directed the actions of Ethiopia's security forces from 2013 to 2015 and one of three officials in charge when those security services embarked on the 'killing' and 'torturing' of Ethiopians. You can see where Tedros is coming from and it's sobering to think that he has been the vehicle for Gates and the Cult to direct the global response to 'Covid'. Think about that. A psychopathic Cult dictates to psychopath Gates who dictates to psychopath Tedros who dictates how countries of the world must respond to a 'Covid virus' never scientifically shown to exist. At the same time psychopathic Cult-owned Silicon Valley information

giants like Google, YouTube, Facebook and Twitter announced very early on that they would give the Cult/Gates/Tedros/WHO version of the narrative free advertising and censor those who challenged their intelligence-insulting, mendacious story.

The next layer in the global 'medical' structure below the Cult, Gates and Tedros are the chief medical officers and science 'advisers' in each of the WHO member countries which means virtually all of them. Medical officers and arbiters of science (they're not) then take the WHO policy and recommended responses and impose them on their country's population while the political 'leaders' say they are deciding policy (they're clearly not) by 'following the science' on the advice of the 'experts' – the same medical officers and science 'advisers' (dictators). In this way with the rarest of exceptions the entire world followed the same policy of lockdown, people distancing, masks and 'vaccines' dictated by the psychopathic Cult, psychopathic Gates and psychopathic Tedros who we are supposed to believe give a damn about the health of the world population they are seeking to enslave. That, amazingly, is all there is to it in terms of crucial decision-making. Medical staff in each country then follow like sheep the dictates of the shepherds at the top of the national medical hierarchies – chief medical officers and science 'advisers' who themselves follow like sheep the shepherds of the World Health Organization and the Cult. Shepherds at the national level often have major funding and other connections to Gates and his Bill and Melinda Gates Foundation which carefully hands out money like confetti at a wedding to control the entire global medical system from the WHO down.

Follow the money

Christopher Whitty, Chief Medical Adviser to the UK Government at the centre of 'virus' policy, a senior adviser to the government's Scientific Advisory Group for Emergencies (SAGE), and Executive Board member of the World Health Organization, was gifted a grant of $40 million by the Bill and Melinda Gates Foundation for malaria research in Africa. The BBC described the unelected Whitty as 'the

official who will probably have the greatest impact on our everyday lives of any individual policymaker in modern times' and so it turned out. What Gates and Tedros have said Whitty has done like his equivalents around the world. Patrick Vallance, co-chair of SAGE and the government's Chief Scientific Adviser, is a former executive of Big Pharma giant GlaxoSmithKline with its fundamental financial and business connections to Bill Gates. In September, 2020, it was revealed that Vallance owned a deferred bonus of shares in GlaxoSmithKline worth £600,000 while the company was 'developing' a 'Covid vaccine'. Move along now – nothing to see here – what could possibly be wrong with that? Imperial College in London, a major player in 'Covid' policy in Britain and elsewhere with its 'Covid-19' Response Team, is funded by Gates and has big connections to China while the now infamous Professor Neil Ferguson, the useless 'computer modeller' at Imperial College is also funded by Gates. Ferguson delivered the dramatically inaccurate excuse for the first lockdowns (much more in the next chapter). The Institute for Health Metrics and Evaluation (IHME) in the United States, another source of outrageously false 'Covid' computer models to justify lockdowns, is bankrolled by Gates who is a vehement promotor of lockdowns. America's version of Whitty and Vallance, the again now infamous Anthony Fauci, has connections to 'Covid vaccine' maker Moderna as does Bill Gates through funding from the Bill and Melinda Gates Foundation. Fauci is director of the National Institute of Allergy and Infectious Diseases (NIAID), a major recipient of Gates money, and they are very close. Deborah Birx who was appointed White House Coronavirus Response Coordinator in February, 2020, is yet another with ties to Gates. Everywhere you look at the different elements around the world behind the coordination and decision making of the 'Covid' hoax there is Bill Gates and his money. They include the World Health Organization; Centers for Disease Control (CDC) in the United States; National Institutes of Health (NIH) of Anthony Fauci; Imperial College and Neil Ferguson; the London School of Hygiene where Chris Whitty worked; Regulatory agencies like the UK Medicines & Healthcare products Regulatory Agency (MHRA)

which gave emergency approval for 'Covid vaccines'; Wellcome Trust; GAVI, the Vaccine Alliance; the Coalition for Epidemic Preparedness Innovations (CEPI); Johns Hopkins University which has compiled the false 'Covid' figures; and the World Economic Forum. A Nationalfile.com article said:

> Gates has a lot of pull in the medical world, he has a multi-million dollar relationship with Dr. Fauci, and Fauci originally took the Gates line supporting vaccines and casting doubt on [the drug hydroxychloroquine]. Coronavirus response team member Dr. Deborah Birx, appointed by former president Obama to serve as United States Global AIDS Coordinator, also sits on the board of a group that has received billions from Gates' foundation, and Birx reportedly used a disputed Bill Gates-funded model for the White House's Coronavirus effort. Gates is a big proponent for a population lockdown scenario for the Coronavirus outbreak.

Another funder of Moderna is the Defense Advanced Research Projects Agency (DARPA), the technology-development arm of the Pentagon and one of the most sinister organisations on earth. DARPA had a major role with the CIA covert technology-funding operation In-Q-Tel in the development of Google and social media which is now at the centre of global censorship. Fauci and Gates are extremely close and openly admit to talking regularly about 'Covid' policy, but then why wouldn't Gates have a seat at every national 'Covid' table after his Foundation committed $1.75 billion to the 'fight against Covid-19'. When passed through our Orwellian Translation Unit this means that he has bought and paid for the Cult-driven 'Covid' response worldwide. Research the major 'Covid' response personnel in your own country and you will find the same Gates funding and other connections again and again. Medical and science chiefs following World Health Organization 'policy' sit atop a medical hierarchy in their country of administrators, doctors and nursing staff. These 'subordinates' are told they must work and behave in accordance with the policy delivered from the 'top' of the national 'health' pyramid which is largely the policy delivered by the WHO which is the policy delivered by Gates and the Cult. The whole 'Covid' narrative has been imposed on medical staff by a climate of fear although great numbers don't even need that to comply. They do so through breathtaking levels of ignorance and

include doctors who go through life simply repeating what Big Pharma and their hierarchical masters tell them to say and believe. No wonder Big Pharma 'medicine' is one of the biggest killers on Planet Earth.

The same top-down system of intimidation operates with regard to the Cult Big Pharma cartel which also dictates policy through national and global medical systems in this way. The Cult and Big Pharma agendas are the same because the former controls and owns the latter. 'Health' administrators, doctors, and nursing staff are told to support and parrot the dictated policy or they will face consequences which can include being fired. How sad it's been to see medical staff meekly repeating and imposing Cult policy without question and most of those who can see through the deceit are only willing to speak anonymously off the record. They know what will happen if their identity is known. This has left the courageous few to expose the lies about the 'virus', face masks, overwhelmed hospitals that aren't, and the dangers of the 'vaccine' that isn't a vaccine. When these medical professionals and scientists, some renowned in their field, have taken to the Internet to expose the truth their articles, comments and videos have been deleted by Cult-owned Facebook, Twitter and YouTube. What a real head-shaker to see YouTube videos with leading world scientists and highly qualified medical specialists with an added link underneath to the notorious Cult propaganda website *Wikipedia* to find the 'facts' about the same subject.

HIV – the 'Covid' trial-run

I'll give you an example of the consequences for health and truth that come from censorship and unquestioning belief in official narratives. The story was told by PCR inventor Kary Mullis in his book *Dancing Naked in the Mind Field*. He said that in 1984 he accepted as just another scientific fact that Luc Montagnier of France's Pasteur Institute and Robert Gallo of America's National Institutes of Health had independently discovered that a 'retrovirus' dubbed HIV (human immunodeficiency virus) caused AIDS. They

were, after all, Mullis writes, specialists in retroviruses. This is how the medical and science pyramids work. Something is announced or *assumed* and then becomes an everybody-knows-that purely through repetition of the assumption as if it is fact. Complete crap becomes accepted truth with no supporting evidence and only repetition of the crap. This is how a 'virus' that doesn't exist became the 'virus' that changed the world. The HIV-AIDS fairy story became a multi-billion pound industry and the media poured out propaganda terrifying the world about the deadly HIV 'virus' that caused the lethal AIDS. By then Mullis was working at a lab in Santa Monica, California, to detect retroviruses with his PCR test in blood donations received by the Red Cross. In doing so he asked a virologist where he could find a reference for HIV being the cause of AIDS. 'You don't need a reference,' the virologist said … *'Everybody knows it.'* Mullis said he wanted to quote a reference in the report he was doing and he said he felt a little funny about not knowing the source of such an important discovery when everyone else seemed to. The virologist suggested he cite a report by the Centers for Disease Control and Prevention (CDC) on morbidity and mortality. Mullis read the report, but it only said that an organism had been identified and did not say how. The report did not identify the original scientific work. Physicians, however, *assumed* (key recurring theme) that if the CDC was convinced that HIV caused AIDS then proof must exist. Mullis continues:

I did computer searches. Neither Montagnier, Gallo, nor anyone else had published papers describing experiments which led to the conclusion that HIV probably caused AIDS. I read the papers in Science for which they had become well known as AIDS doctors, but all they had said there was that they had found evidence of a past infection by something which was probably HIV in some AIDS patients.

They found antibodies. Antibodies to viruses had always been considered evidence of past disease, not present disease. Antibodies signaled that the virus had been defeated. The patient had saved himself. There was no indication in these papers that this virus caused a disease. They didn't show that everybody with the antibodies had the disease. In fact they found some healthy people with antibodies.

Mullis asked why their work had been published if Montagnier and Gallo hadn't really found this evidence, and why had they been fighting so hard to get credit for the discovery? He says he was hesitant to write 'HIV is the probable cause of AIDS' until he found published evidence to support that. 'Tens of thousands of scientists and researchers were spending billions of dollars a year doing research based on this idea,' Mullis writes. 'The reason had to be there somewhere; otherwise these people would not have allowed their research to settle into one narrow channel of investigation.' He said he lectured about PCR at numerous meetings where people were always talking about HIV and he asked them how they knew that HIV was the cause of AIDS:

> Everyone said something. Everyone had the answer at home, in the office, in some drawer. They all knew, and they would send me the papers as soon as they got back. But I never got any papers. Nobody ever sent me the news about how AIDS was caused by HIV.

Eventually Mullis was able to ask Montagnier himself about the reference proof when he lectured in San Diego at the grand opening of the University of California AIDS Research Center. Mullis says this was the last time he would ask his question without showing anger. Montagnier said he should reference the CDC report. 'I read it', Mullis said, and it didn't answer the question. 'If Montagnier didn't know the answer who the hell did?' Then one night Mullis was driving when an interview came on National Public Radio with Peter Duesberg, a prominent virologist at Berkeley and a California Scientist of the Year. Mullis says he finally understood why he could not find references that connected HIV to AIDS – *there weren't any!* No one had ever proved that HIV causes AIDS even though it had spawned a multi-billion pound global industry and the media was repeating this as fact every day in their articles and broadcasts terrifying the shit out of people about AIDS and giving the impression that a positive test for HIV (see 'Covid') was a death sentence. Duesberg was a threat to the AIDS gravy train and the agenda that underpinned it. He was therefore abused and castigated after he told the Proceedings of the National Academy of Sciences

there was no good evidence implicating the new 'virus'. Editors rejected his manuscripts and his research funds were deleted. Mullis points out that the CDC has defined AIDS as one of more than 30 diseases *if accompanied* by a positive result on a test that detects antibodies to HIV; but those same diseases are not defined as AIDS cases when antibodies are not detected:

> If an HIV-positive woman develops uterine cancer, for example, she is considered to have AIDS. If she is not HIV positive, she simply has uterine cancer. An HIV-positive man with tuberculosis has AIDS; if he tests negative he simply has tuberculosis. If he lives in Kenya or Colombia, where the test for HIV antibodies is too expensive, he is simply presumed to have the antibodies and therefore AIDS, and therefore he can be treated in the World Health Organization's clinic. It's the only medical help available in some places. And it's free, because the countries that support WHO are worried about AIDS.

Mullis accuses the CDC of continually adding new diseases (see ever more 'Covid symptoms') to the grand AIDS definition and of virtually doctoring the books to make it appear as if the disease continued to spread. He cites how in 1993 the CDC enormously broadened its AIDS definition and county health authorities were delighted because they received $2,500 per year from the Federal government for every reported AIDS case. Ladies and gentlemen, I have just described, via Kary Mullis, the 'Covid pandemic' of 2020 and beyond. Every element is the same and it's been pulled off in the same way by the same networks.

The 'Covid virus' exists? Okay – prove it. Er … still waiting

What Kary Mullis described with regard to 'HIV' has been repeated with 'Covid'. A claim is made that a new, or 'novel', infection has been found and the entire medical system of the world repeats that as fact exactly as they did with HIV and AIDS. No one in the mainstream asks rather relevant questions such as 'How do you know?' and 'Where is your proof?' The SARS-Cov-2 'virus' and the 'Covid-19 disease' became an overnight 'everybody-knows-that'. The origin could be debated and mulled over, but what you could not suggest was that 'SARS-Cov-2' didn't exist. That would be

ridiculous. 'Everybody knows' the 'virus' exists. Well, I didn't for one along with American proper doctors like Andrew Kaufman and Tom Cowan and long-time American proper journalist Jon Rappaport. We dared to pursue the obvious and simple question: 'Where's the evidence?' The overwhelming majority in medicine, journalism and the general public did not think to ask that. After all, *everyone knew* there was a new 'virus'. Everyone was saying so and I heard it on the BBC. Some would eventually argue that the 'deadly virus' was nothing like as deadly as claimed, but few would venture into the realms of its very existence. Had they done so they would have found that the evidence for that claim had gone AWOL as with HIV causes AIDS. In fact, not even that. For something to go AWOL it has to exist in the first place and scientific proof for a 'SARS-Cov-2' can be filed under nothing, nowhere and zilch.

Dr Andrew Kaufman is a board-certified forensic psychiatrist in New York State, a Doctor of Medicine and former Assistant Professor and Medical Director of Psychiatry at SUNY Upstate Medical University, and Medical Instructor of Hematology and Oncology at the Medical School of South Carolina. He also studied biology at the Massachusetts Institute of Technology (MIT) and trained in Psychiatry at Duke University. Kaufman is retired from allopathic medicine, but remains a consultant and educator on natural healing, I saw a video of his very early on in the 'Covid' hoax in which he questioned claims about the 'virus' in the absence of any supporting evidence and with plenty pointing the other way. I did everything I could to circulate his work which I felt was asking the pivotal questions that needed an answer. I can recommend an excellent pull-together interview he did with the website The Last Vagabond entitled *Dr Andrew Kaufman: Virus Isolation, Terrain Theory and Covid-19* and his website is andrewkaufmanmd.com. Kaufman is not only a forensic psychiatrist; he is forensic in all that he does. He always reads original scientific papers, experiments and studies instead of second-third-fourth-hand reports about the 'virus' in the media which are repeating the repeated repetition of the narrative. When he did so with the original Chinese 'virus' papers Kaufman

realised that there was no evidence of a 'SARS-Cov-2'. They had never – from the start – shown it to exist and every repeat of this claim worldwide was based on the accepted existence of proof that was nowhere to be found – see Kary Mullis and HIV. Here we go again.

Let's postulate

Kaufman discovered that the Chinese authorities immediately concluded that the cause of an illness that broke out among about 200 initial patients in Wuhan was a 'new virus' when there were no grounds to make that conclusion. The alleged 'virus' was not isolated from other genetic material in their samples and then shown through a system known as Koch's postulates to be the causative agent of the illness. The world was told that the SARS-Cov-2 'virus' caused a disease they called 'Covid-19' which had 'flu-like' symptoms and could lead to respiratory problems and pneumonia. If it wasn't so tragic it would almost be funny. *'Flu-like' symptoms'? Pneumonia? Respiratory disease?* What in *CHINA* and particularly in *Wuhan*, one of the most polluted cities in the world with a resulting epidemic of respiratory disease?? Three hundred thousand people get pneumonia in China every year and there are nearly a billion cases worldwide of 'flu-like symptoms'. These have a whole range of causes – including pollution in Wuhan – but no other possibility was credibly considered in late 2019 when the world was told there was a new and deadly 'virus'. The global prevalence of pneumonia and 'flu-like systems' gave the Cult networks unlimited potential to re-diagnose these other causes as the mythical 'Covid-19' and that is what they did from the very start. Kaufman revealed how Chinese medical and science authorities (all subordinates to the Cult-owned communist government) took genetic material from the lungs of only a few of the first patients. The material contained their own cells, bacteria, fungi and other microorganisms living in their bodies. The only way you could prove the existence of the 'virus' and its responsibility for the alleged 'Covid-19' was to isolate the virus from all the other material – a process also known as 'purification' – and

then follow the postulates sequence developed in the late 19th century by German physician and bacteriologist Robert Koch which became the 'gold standard' for connecting an alleged causation agent to a disease:

1. The microorganism (bacteria, fungus, virus, etc.) must be present in every case of the disease and all patients must have the same symptoms. It must also *not be present in healthy individuals*.

2. The microorganism must be isolated from the host with the disease. If the microorganism is a bacteria or fungus it must be grown in a pure culture. If it is a virus, it must be purified (i.e. containing no other material except the virus particles) from a clinical sample.

3. The specific disease, with all of its characteristics, must be reproduced when the infectious agent (the purified virus or a pure culture of bacteria or fungi) is inoculated into a healthy, susceptible host.

4. The microorganism must be recoverable from the experimentally infected host as in step 2.

Not one of these criteria has been met in the case of 'SARS-Cov-2' and 'Covid-19'. Not ONE. *EVER*. Robert Koch refers to bacteria and not viruses. What are called 'viral particles' are so minute (hence masks are useless by any definition) that they could only be seen after the invention of the electron microscope in the 1930s and can still only be observed through that means. American bacteriologist and virologist Thomas Milton Rivers, the so-called 'Father of Modern Virology' who was very significantly director of the Rockefeller Institute for Medical Research in the 1930s, developed a less stringent version of Koch's postulates to identify 'virus' causation known as 'Rivers criteria'. 'Covid' did not pass that process either. Some even doubt whether any 'virus' can be isolated from other particles containing genetic material in the Koch method. Freedom of Information requests in many countries asking for scientific proof that the 'Covid virus' has been purified and isolated and shown to exist have all come back with a 'we don't have that' and when this happened with a request to the UK Department of Health they added this comment:

However, outside of the scope of the [Freedom of Information Act] and on a discretionary basis, the following information has been advised to us, which may be of interest. Most infectious diseases are caused by viruses, bacteria or fungi. Some bacteria or fungi have the capacity to grow on their own in isolation, for example in colonies on a petri dish. Viruses are different in that they are what we call 'obligate pathogens' – that is, they cannot survive or reproduce without infecting a host ...

… For some diseases, it is possible to establish causation between a microorganism and a disease by isolating the pathogen from a patient, growing it in pure culture and reintroducing it to a healthy organism. These are known as 'Koch's postulates' and were developed in 1882. However, as our understanding of disease and different disease-causing agents has advanced, these are no longer the method for determining causation [Andrew Kaufman asks why in that case are there two published articles falsely claiming to satisfy Koch's postulates].

It has long been known that viral diseases cannot be identified in this way as viruses cannot be grown in 'pure culture'. When a patient is tested for a viral illness, this is normally done by looking for the presence of antigens, or viral genetic code in a host with molecular biology techniques [Kaufman asks how you could know the origin of these chemicals without having a pure culture for comparison].

For the record 'antigens' are defined so:

Invading microorganisms have antigens on their surface that the human body can recognise as being foreign – meaning not belonging to it. When the body recognises a foreign antigen, lymphocytes (white blood cells) produce antibodies, which are complementary in shape to the antigen.

Notwithstanding that this is open to question in relation to 'SARS-Cov-2' the presence of 'antibodies' can have many causes and they are found in people that are perfectly well. Kary Mullis said: 'Antibodies … had always been considered evidence of past disease, not present disease.'

'Covid' really is a *computer* 'virus'

Where the UK Department of Health statement says 'viruses' are now 'diagnosed' through a 'viral genetic code in a host with molecular biology techniques', they mean … *the PCR test* which its inventor said cannot test for infectious disease. They have no credible method of connecting a 'virus' to a disease and we will see that there is no scientific proof that any 'virus' causes any disease or there is any such thing as a 'virus' in the way that it is described. Tenacious Canadian researcher Christine Massey and her team made

some 40 Freedom of Information requests to national public health agencies in different countries asking for proof that SARS-CoV-2 has been isolated and not one of them could supply that information. Massey said of her request in Canada: 'Freedom of Information reveals Public Health Agency of Canada has no record of 'SARS-COV-2' isolation performed by anyone, anywhere, ever.' If you accept the comment from the UK Department of Health it's because they can't isolate a 'virus'. Even so many 'science' papers claimed to have isolated the 'Covid virus' until they were questioned and had to admit they hadn't. A reply from the Robert Koch Institute in Germany was typical: 'I am not aware of a paper which purified isolated SARS-CoV-2.' So what the hell was Christian Drosten and his gang using to design the 'Covid' testing protocol that has produced all the illusory Covid' cases and 'Covid' deaths when the head of the Chinese version of the CDC admitted there was a problem right from the start in that the 'virus' had never been isolated/purified? Breathe deeply: What they are calling 'Covid' is actually created by a *computer program* i.e. *they made it up* – er, that's it. They took lung fluid, with many sources of genetic material, from one single person alleged to be infected with Covid-19 by a PCR test which they *claimed*, without clear evidence, contained a 'virus'. They used several computer programs to create a model of a theoretical virus genome sequence from more than fifty-six million small sequences of RNA, each of an unknown source, assembling them like a puzzle with no known solution. The computer filled in the gaps with sequences from bits in the gene bank to make it look like a bat SARS-like coronavirus! A wave of the magic wand and poof, an in silico (computer-generated) genome, a scientific fantasy, was created. UK health researcher Dr Kevin Corbett made the same point with this analogy:

> … It's like giving you a few bones and saying that's your fish. It could be any fish. Not even a skeleton. Here's a few fragments of bones. That's your fish … It's all from gene bank and the bits of the virus sequence that weren't there they made up.
>
> They synthetically created them to fill in the blanks. That's what genetics is; it's a code. So it's ABBBCCDDD and you're missing some what you think is EEE so you put it in. It's all

synthetic. You just manufacture the bits that are missing. This is the end result of the geneticization of virology. This is basically a computer virus.

Further confirmation came in an email exchange between British citizen journalist Frances Leader and the government's Medicines & Healthcare Products Regulatory Agency (the Gates-funded MHRA) which gave emergency permission for untested 'Covid vaccines' to be used. The agency admitted that the 'vaccine' is not based on an isolated 'virus', but comes from a *computer-generated model*. Frances Leader was naturally banned from Cult-owned fascist Twitter for making this exchange public. The process of creating computer-generated alleged 'viruses' is called 'in silico' or 'in silicon' – computer chips – and the term 'in silico' is believed to originate with biological experiments using only a computer in 1989. 'Vaccines' involved with 'Covid' are also produced 'in silico' or by computer not a natural process. If the original 'virus' is nothing more than a made-up computer model how can there be 'new variants' of something that never existed in the first place? They are not new 'variants'; they are new *computer models* only minutely different to the original program and designed to further terrify the population into having the 'vaccine' and submitting to fascism. You want a 'new variant'? Click, click, enter – there you go. Tell the medical profession that you have discovered a 'South African variant', 'UK variants' or a 'Brazilian variant' and in the usual HIV-causes-AIDS manner they will unquestioningly repeat it with no evidence whatsoever to support these claims. They will go on television and warn about the dangers of 'new variants' while doing nothing more than repeating what they have been told to be true and knowing that any deviation from that would be career suicide. Big-time insiders will know it's a hoax, but much of the medical community is clueless about the way they are being played and themselves play the public without even being aware they are doing so. What an interesting 'coincidence' that AstraZeneca and Oxford University were conducting 'Covid vaccine trials' in the three countries – the UK, South Africa and Brazil – where the first three 'variants' were claimed to have 'broken out'.

Here's your 'virus' – it's a unicorn

Dr Andrew Kaufman presented a brilliant analysis describing how the 'virus' was imagined into fake existence when he dissected an article published by *Nature* and written by 19 authors detailing *alleged* 'sequencing of a complete viral genome' of the 'new SARS-CoV-2 virus'. This computer-modelled *in silico* genome was used as a template for all subsequent genome sequencing experiments that resulted in the so-called variants which he said now number more than *6,000*. The fake genome was constructed from more than 56 million individual short strands of RNA. Those little pieces were assembled into longer pieces by finding areas of overlapping sequences. The computer programs created over two million possible combinations from which the authors simply chose the longest one. They then compared this to a 'bat virus' and the computer 'alignment' rearranged the sequence and filled in the gaps! They called this computer-generated abomination the 'complete genome'. Dr Tom Cowan, a fellow medical author and collaborator with Kaufman, said such computer-generation constitutes scientific fraud and he makes this superb analogy:

> Here is an equivalency: A group of researchers claim to have found a unicorn because they found a piece of a hoof, a hair from a tail, and a snippet of a horn. They then add that information into a computer and program it to re-create the unicorn, and they then claim this computer re-creation is the real unicorn. Of course, they had never actually seen a unicorn so could not possibly have examined its genetic makeup to compare their samples with the actual unicorn's hair, hooves and horn.
>
> The researchers claim they decided which is the real genome of SARS-CoV-2 by 'consensus', sort of like a vote. Again, different computer programs will come up with different versions of the imaginary 'unicorn', so they come together as a group and decide which is the real imaginary unicorn.

This is how the 'virus' that has transformed the world was brought into fraudulent 'existence'. Extraordinary, yes, but as the Nazis said the bigger the lie the more will believe it. Cowan, however, wasn't finished and he went on to identify what he called the real blockbuster in the paper. He quotes this section from a paper written

by virologists and published by the CDC and then explains what it means:

> Therefore, we examined the capacity of SARS-CoV-2 to infect and replicate in several common primate and human cell lines, including human adenocarcinoma cells (A549), human liver cells (HUH 7.0), and human embryonic kidney cells (HEK-293T). In addition to Vero E6 and Vero CCL81 cells. ... Each cell line was inoculated at high multiplicity of infection and examined 24h post-infection.
>
> No CPE was observed in any of the cell lines except in Vero cells, which grew to greater than 10 to the 7th power at 24 h post-infection. In contrast, HUH 7.0 and 293T showed only modest viral replication, and A549 cells were incompatible with SARS CoV-2 infection.

Cowan explains that when virologists attempt to prove infection they have three possible 'hosts' or models on which they can test. The first was humans. Exposure to humans was generally not done for ethical reasons and has never been done with SARS-CoV-2 or any coronavirus. The second possible host was animals. Cowan said that forgetting for a moment that they never actually use purified virus when exposing animals they do use solutions that they *claim* contain the virus. Exposure to animals has been done with SARS-CoV-2 in an experiment involving mice and this is what they found: *None of the wild (normal) mice got sick*. In a group of genetically-modified mice, a statistically insignificant number lost weight and had slightly bristled fur, but they experienced nothing like the illness called 'Covid-19'. Cowan said the third method – the one they mostly rely on – is to inoculate solutions they *say* contain the virus onto a variety of tissue cultures. This process had never been shown to kill tissue *unless* the sample material was starved of nutrients and poisoned as *part of the process*. Yes, incredibly, in tissue experiments designed to show the 'virus' is responsible for killing the tissue they starve the tissue of nutrients and add toxic drugs including antibiotics and they do not have control studies to see if it's the starvation and poisoning that is degrading the tissue rather than the 'virus' they allege to be in there somewhere. You want me to pinch you? Yep, I understand. Tom Cowan said this about the whole nonsensical farce as he explains what that quote from the CDC paper really means:

The shocking thing about the above quote is that using their own methods, the virologists found that solutions containing SARS-CoV-2 – even in high amounts – were NOT, I repeat NOT, infective to any of the three human tissue cultures they tested. In plain English, this means they proved, on their terms, that this 'new coronavirus' is not infectious to human beings. It is ONLY infective to monkey kidney cells, and only then when you add two potent drugs (gentamicin and amphotericin), known to be toxic to kidneys, to the mix.

My friends, read this again and again. These virologists, published by the CDC, performed a clear proof, on their terms, showing that the SARS-CoV-2 virus is harmless to human beings. That is the only possible conclusion, but, unfortunately, this result is not even mentioned in their conclusion. They simply say they can provide virus stocks cultured only on monkey Vero cells, thanks for coming.

Cowan concluded: 'If people really understood how this "science" was done, I would hope they would storm the gates and demand honesty, transparency and truth.' Dr Michael Yeadon, former Vice President and Chief Scientific Adviser at drug giant Pfizer has been a vocal critic of the 'Covid vaccine' and its potential for multiple harm. He said in an interview in April, 2021, that 'not one [vaccine] has the virus. He was asked why vaccines normally using a 'dead' version of a disease to activate the immune system were not used for 'Covid' and instead we had the synthetic methods of the 'mRNA Covid vaccine'. Yeadon said that to do the former 'you'd have to have some of [the virus] wouldn't you?' He added: 'No-one's got any – seriously.' Yeadon said that surely they couldn't have fooled the whole world for a year without having a virus, 'but oddly enough ask around – no one's got it'. He didn't know why with all the 'great labs' around the world that the virus had not been isolated – 'Maybe they've been too busy running bad PCR tests and vaccines that people don't need.' What is today called 'science' is not 'science' at all. Science is no longer what is, but whatever people can be manipulated to *believe* that it is. Real science has been hijacked by the Cult to dispense and produce the 'expert scientists' and contentions that suit the agenda of the Cult. How big-time this has happened with the 'Covid' hoax which is entirely based on fake science delivered by fake 'scientists' and fake 'doctors'. The human-caused climate change hoax is also entirely based on fake science delivered by fake 'scientists' and fake 'climate experts'. In both cases real

scientists, climate experts and doctors have their views suppressed and deleted by the Cult-owned science establishment, media and Silicon Valley. This is the 'science' that politicians claim to be 'following' and a common denominator of 'Covid' and climate are Cult psychopaths Bill Gates and his mate Klaus Schwab at the Gates-funded World Economic Forum. But, don't worry, it's all just a coincidence and absolutely nothing to worry about. *Zzzzzzzz*.

What is a 'virus' REALLY?

Dr Tom Cowan is one of many contesting the very existence of viruses let alone that they cause disease. This is understandable when there is no scientific evidence for a disease-causing 'virus'. German virologist Dr Stefan Lanka won a landmark case in 2017 in the German Supreme Court over his contention that there is no such thing as a measles virus. He had offered a big prize for anyone who could prove there is and Lanka won his case when someone sought to claim the money. There is currently a prize of more than 225,000 euros on offer from an Isolate Truth Fund for anyone who can prove the isolation of SARS-CoV-2 and its genetic substance. Lanka wrote in an article headed 'The Misconception Called Virus' that scientists think a 'virus' is causing tissue to become diseased and degraded when in fact it is the *processes they are using* which do that – not a 'virus'. Lanka has done an important job in making this point clear as Cowan did in his analysis of the CDC paper. Lanka says that all claims about viruses as disease-causing pathogens are wrong and based on 'easily recognisable, understandable and verifiable misinterpretations.' Scientists believed they were working with 'viruses' in their laboratories when they were really working with 'typical particles of specific dying tissues or cells …' Lanka said that the tissue decaying process claimed to be caused by a 'virus' still happens when no alleged 'virus' is involved. It's the *process* that does the damage and not a 'virus'. The genetic sample is deprived of nutrients, removed from its energy supply through removal from the body and then doused in toxic antibiotics to remove any bacteria. He confirms again that establishment scientists do not (pinch me)

conduct control experiments to see if this is the case and if they did they would see the claims that 'viruses' are doing the damage is nonsense. He adds that during the measles 'virus' court case he commissioned an independent laboratory to perform just such a control experiment and the result was that the tissues and cells died in the exact same way as with alleged 'infected' material. This is supported by a gathering number of scientists, doctors and researchers who reject what is called 'germ theory' or the belief in the body being infected by contagious sources emitted by other people. Researchers Dawn Lester and David Parker take the same stance in their highly-detailed and sourced book *What Really Makes You Ill – Why everything you thought you knew about disease is wrong* which was recommended to me by a number of medical professionals genuinely seeking the truth. Lester and Parker say there is no provable scientific evidence to show that a 'virus' can be transmitted between people or people and animals or animals and people:

> The definition also claims that viruses are the cause of many diseases, as if this has been definitively proven. But this is not the case; there is no original scientific evidence that definitively demonstrates that any virus is the cause of any disease. The burden of proof for any theory lies with those who proposed it; but none of the existing documents provides 'proof' that supports the claim that 'viruses' are pathogens.

Dr Tom Cowan employs one of his clever analogies to describe the process by which a 'virus' is named as the culprit for a disease when what is called a 'virus' is only material released by cells detoxing themselves from infiltration by chemical or radiation poisoning. The tidal wave of technologically-generated radiation in the 'smart' modern world plus all the toxic food and drink are causing this to happen more than ever. Deluded 'scientists' misread this as a gathering impact of what they wrongly label 'viruses'.

Paper can infect houses

Cowan said in an article for davidicke.com – with his tongue only mildly in his cheek – that he believed he had made a tremendous

discovery that may revolutionise science. He had discovered that small bits of paper are alive, 'well alive-ish', can 'infect' houses, and then reproduce themselves inside the house. The result was that this explosion of growth in the paper inside the house causes the house to explode, blowing it to smithereens. His evidence for this new theory is that in the past months he had carefully examined many of the houses in his neighbourhood and found almost no scraps of paper on the lawns and surrounds of the house. There was an occasional stray label, but nothing more. Then he would return to these same houses a week or so later and with a few, not all of them, particularly the old and decrepit ones, he found to his shock and surprise they were littered with stray bits of paper. He knew then that the paper had infected these houses, made copies of itself, and blew up the house. A young boy on a bicycle at one of the sites told him he had seen a demolition crew using dynamite to explode the house the previous week, but Cowan dismissed this as the idle thoughts of silly boys because 'I was on to something big'. He was on to how 'scientists' mistake genetic material in the detoxifying process for something they call a 'virus'. Cowan said of his house and paper story:

If this sounds crazy to you, it's because it should. This scenario is obviously nuts. But consider this admittedly embellished, for effect, current viral theory that all scientists, medical doctors and virologists currently believe.

He takes the example of the 'novel SARS-Cov2' virus to prove the point. First they take someone with an undefined illness called 'Covid-19' and don't even attempt to find any virus in their sputum. Never mind the scientists still describe how this 'virus', which they have not located attaches to a cell receptor, injects its genetic material, in 'Covid's' case, RNA, into the cell. The RNA once inserted exploits the cell to reproduce itself and makes 'thousands, nay millions, of copies of itself … Then it emerges victorious to claim its next victim':

If you were to look in the scientific literature for proof, actual scientific proof, that uniform SARS-CoV2 viruses have been properly isolated from the sputum of a sick person, that actual spike proteins could be seen protruding from the virus (which has not been found), you would find that such evidence doesn't exist.

If you go looking in the published scientific literature for actual pictures, proof, that these spike proteins or any viral proteins are ever attached to any receptor embedded in any cell membrane, you would also find that no such evidence exists. If you were to look for a video or documented evidence of the intact virus injecting its genetic material into the body of the cell, reproducing itself and then emerging victorious by budding off the cell membrane, you would find that no such evidence exists.

The closest thing you would find is electron micrograph pictures of cellular particles, possibly attached to cell debris, both of which to be seen were stained by heavy metals, a process that completely distorts their architecture within the living organism. This is like finding bits of paper stuck to the blown-up bricks, thereby proving the paper emerged by taking pieces of the bricks on its way out.

The Enders baloney

Cowan describes the 'Covid' story as being just as make-believe as his paper story and he charts back this fantasy to a Nobel Prize winner called John Enders (1897-1985), an American biomedical scientist who has been dubbed 'The Father of Modern Vaccines'. Enders is claimed to have 'discovered' the process of the viral culture which 'proved' that a 'virus' caused measles. Cowan explains how Enders did this 'by using the EXACT same procedure that has been followed by every virologist to find and characterize every new virus since 1954'. Enders took throat swabs from children with measles and immersed them in 2ml of milk. Penicillin (100u/ml) and the antibiotic streptomycin (50,g/ml) were added and the whole mix was centrifuged – rotated at high speed to separate large cellular debris from small particles and molecules as with milk and cream, for example. Cowan says that if the aim is to find little particles of genetic material ('viruses') in the snot from children with measles it would seem that the last thing you would do is mix the snot with other material – milk –that also has genetic material. 'How are you ever going to know whether whatever you found came from the snot or the milk?' He points out that streptomycin is a 'nephrotoxic' or poisonous-to-the-kidney drug. You will see the relevance of that

shortly. Cowan says that it gets worse, much worse, when Enders describes the culture medium upon which the virus 'grows': 'The culture medium consisted of bovine amniotic fluid (90%), beef embryo extract (5%), horse serum (5%), antibiotics and phenol red as an indicator of cell metabolism.' Cowan asks incredulously: 'Did he just say that the culture medium also contained fluids and tissues that are themselves rich sources of genetic material?' The genetic cocktail, or 'medium', is inoculated onto tissue and cells from rhesus monkey *kidney* tissue. This is where the importance of streptomycin comes in and currently-used antimicrobials and other drugs that are *poisonous to kidneys* and used in ALL modern viral cultures (e.g. gentamicin, streptomycin, and amphotericin). Cowan asks: 'How are you ever going to know from this witch's brew where any genetic material comes from as we now have five different sources of rich genetic material in our mix?' Remember, he says, that all genetic material, whether from monkey kidney tissues, bovine serum, milk, etc., is made from the exact same components. The same central question returns: 'How are you possibly going to know that it was the virus that killed the kidney tissue and not the toxic antibiotic and starvation rations on which you are growing the tissue?' John Enders answered the question himself – *you can't*:

A second agent was obtained from an uninoculated culture of monkey kidney cells. The cytopathic changes [death of the cells] it induced in the unstained preparations could not be distinguished with confidence from the viruses isolated from measles.

The death of the cells ('cytopathic changes') happened in exactly the same manner, whether they inoculated the kidney tissue with the measles snot or not, Cowan says. 'This is evidence that the destruction of the tissue, the very proof of viral causation of illness, was not caused by anything in the snot because they saw the same destructive effect when the snot was not even used … the cytopathic, i.e., cell-killing, changes come from the process of the culture itself, not from any virus in any snot, period.' Enders quotes in his 1957 paper a virologist called Ruckle as reporting similar findings 'and in addition has isolated an agent from monkey kidney tissue that is so

far indistinguishable from human measles virus'. In other words, Cowan says, these particles called 'measles viruses' are simply and clearly breakdown products of the starved and poisoned tissue. For measles 'virus' see all 'viruses' including the so-called 'Covid virus'. Enders, the 'Father of Modern Vaccines', also said:

There is a potential risk in employing cultures of primate cells for the production of vaccines composed of attenuated virus, since the presence of other agents possibly latent in primate tissues cannot be definitely excluded by any known method.

Cowan further quotes from a paper published in the journal *Viruses* in May, 2020, while the 'Covid pandemic' was well underway in the media if not in reality. 'EVs' here refers to particles of genetic debris from our own tissues, such as exosomes of which more in a moment: 'The remarkable resemblance between EVs and viruses has caused quite a few problems in the studies focused on the analysis of EVs released during viral infections.' Later the paper adds that to date a reliable method that can actually guarantee a complete separation (of EVs from viruses) DOES NOT EXIST. This was published at a time when a fairy tale 'virus' was claimed in total certainty to be causing a fairy tale 'viral disease' called 'Covid-19' – a fairy tale that was already well on the way to transforming human society in the image that the Cult has worked to achieve for so long. Cowan concludes his article:

To summarize, there is no scientific evidence that pathogenic viruses exist. What we think of as 'viruses' are simply the normal breakdown products of dead and dying tissues and cells. When we are well, we make fewer of these particles; when we are starved, poisoned, suffocated by wearing masks, or afraid, we make more.

There is no engineered virus circulating and making people sick. People in laboratories all over the world are making genetically modified products to make people sick. These are called vaccines. There is no virome, no 'ecosystem' of viruses, viruses are not 8%, 50% or 100 % of our genetic material. These are all simply erroneous ideas based on the misconception called a virus.

What is 'Covid'? Load of bollocks

The background described here by Cowan and Lanka was emphasised in the first video presentation that I saw by Dr Andrew Kaufman when he asked whether the 'Covid virus' was in truth a natural defence mechanism of the body called 'exosomes'. These are released by cells when in states of toxicity – see the same themes returning over and over. They are released ever more profusely as chemical and radiation toxicity increases and think of the potential effect therefore of 5G alone as its destructive frequencies infest the human energetic information field with a gathering pace (5G went online in Wuhan in 2019 as the 'virus' emerged). I'll have more about this later. Exosomes transmit a warning to the rest of the body that 'Houston, we have a problem'. Kaufman presented images of exosomes and compared them with 'Covid' under an electron microscope and the similarity was remarkable. They both attach to the same cell receptors (*claimed* in the case of 'Covid'), contain the same genetic material in the form of RNA or ribonucleic acid, and both are found in 'viral cell cultures' with damaged or dying cells. James Hildreth MD, President and Chief Executive Officer of the Meharry Medical College at Johns Hopkins, said: 'The virus is fully an exosome in every sense of the word.' Kaufman's conclusion was that there is no 'virus': 'This entire pandemic is a completely manufactured crisis … there is no evidence of anyone dying from [this] illness.' Dr Tom Cowan and Sally Fallon Morell, authors of *The Contagion Myth*, published a statement with Dr Kaufman in February, 2021, explaining why the 'virus' does not exist and you can read it that in full in the Appendix.

'Virus' theory can be traced to the 'cell theory' in 1858 of German physician Rudolf Virchow (1821-1920) who contended that disease originates from a single cell infiltrated by a 'virus'. Dr Stefan Lanka said that findings and insights with respect to the structure, function and central importance of tissues in the creation of life, which were already known in 1858, comprehensively refute the cell theory. Virchow ignored them. We have seen the part later played by John Enders in the 1950s and Lanka notes that infection theories were only established as a global dogma through the policies and

eugenics of the Third Reich in Nazi Germany (creation of the same Sabbatian cult behind the 'Covid' hoax). Lanka said: 'Before 1933, scientists dared to contradict this theory; after 1933, these critical scientists were silenced'. Dr Tom Cowan's view is that ill-heath is caused by too much of something, too little of something, or toxification from chemicals and radiation – not contagion. We must also highlight as a major source of the 'virus' theology a man still called the 'Father of Modern Virology' – Thomas Milton Rivers (1888-1962). There is no way given the Cult's long game policy that it was a coincidence for the 'Father of Modern Virology' to be director of the Rockefeller Institute for Medical Research from 1937 to 1956 when he is credited with making the Rockefeller Institute a leader in 'viral research'. Cult Rockefellers were the force behind the creation of Big Pharma 'medicine', established the World Health Organisation in 1948, and have long and close associations with the Gates family that now runs the WHO during the pandemic hoax through mega-rich Cult gofer and psychopath Bill Gates.

Only a Renegade Mind can see through all this bullshit by asking the questions that need to be answered, not taking 'no' or prevarication for an answer, and certainly not hiding from the truth in fear of speaking it. Renegade Minds have always changed the world for the better and they will change this one no matter how bleak it may currently appear to be.

CHAPTER SIX

Sequence of deceit

If you tell the truth, you don't have to remember anything
Mark Twain

Against the background that I have laid out this far the sequence that took us from an invented 'virus' in Cult-owned China in late 2019 to the fascist transformation of human society can be seen and understood in a whole new context.

We were told that a deadly disease had broken out in Wuhan and the world media began its campaign (coordinated by behavioural psychologists as we shall see) to terrify the population into unquestioning compliance. We were shown images of Chinese people collapsing in the street which never happened in the West with what was supposed to be the same condition. In the earliest days when alleged cases and deaths were few the fear register was hysterical in many areas of the media and this would expand into the common media narrative across the world. The real story was rather different, but we were never told that. The Chinese government, one of the Cult's biggest centres of global operation, said they had discovered a new illness with flu-like and pneumonia-type symptoms in a city with such toxic air that it is overwhelmed with flu-like symptoms, pneumonia and respiratory disease. Chinese scientists said it was a new – 'novel' – coronavirus which they called Sars-Cov-2 and that it caused a disease they labelled 'Covid-19'. There was no evidence for this and the 'virus' has never to this day been isolated, purified and its genetic code established from that. It

was from the beginning a computer-generated fiction. Stories of Chinese whistleblowers saying the number of deaths was being supressed or that the 'new disease' was related to the Wuhan bio-lab misdirected mainstream and alternative media into cul-de-sacs to obscure the real truth – there was no 'virus'.

Chinese scientists took genetic material from the lung fluid of just a few people and said they had found a 'new' disease when this material had a wide range of content. There was no evidence for a 'virus' for the very reasons explained in the last two chapters. The 'virus' has never been shown to (a) exist and (b) cause any disease. People were diagnosed on symptoms that are so widespread in Wuhan and polluted China and with a PCR test that can't detect infectious disease. On this farce the whole global scam was sold to the rest of the world which would also diagnose respiratory disease as 'Covid-19' from symptoms alone or with a PCR test not testing for a 'virus'. Flu miraculously disappeared *worldwide* in 2020 and into 2021 as it was redesignated 'Covid-19'. It was really the same old flu with its 'flu-like' symptoms attributed to 'flu-like' 'Covid-19'. At the same time with very few exceptions the Chinese response of draconian lockdown and fascism was the chosen weapon to respond across the West as recommended by the Cult-owned Tedros at the Cult-owned World Health Organization run by the Cult-owned Gates. All was going according to plan. Chinese scientists – everything in China is controlled by the Cult-owned government – compared their contaminated RNA lung-fluid material with other RNA sequences and said it appeared to be just under 80 percent identical to the SARS-CoV-1 'virus' claimed to be the cause of the SARS (severe acute respiratory syndrome) 'outbreak' in 2003. They decreed that because of this the 'new virus' had to be related and they called it SARS-CoV-2. There are some serious problems with this assumption and *assumption* was all it was. Most 'factual' science turns out to be assumptions repeated into everyone-knows-that. A match of under 80-percent is meaningless. Dr Kaufman makes the point that there's a *96 percent* genetic correlation between humans and chimpanzees, but 'no one would say our genetic material is part

of the chimpanzee family'. Yet the Chinese authorities were claiming that a much lower percentage, less than 80 percent, proved the existence of a new 'coronavirus'. For goodness sake human DNA is 60 percent similar to a *banana*.

You are feeling sleepy

The entire 'Covid' hoax is a global Psyop, a psychological operation to program the human mind into believing and fearing a complete fantasy. A crucial aspect of this was what *appeared* to happen in Italy. It was all very well streaming out daily images of an alleged catastrophe in Wuhan, but to the Western mind it was still on the other side of the world in a very different culture and setting. A reaction of 'this could happen to me and my family' was still nothing like as intense enough for the mind-doctors. The Cult needed a Western example to push people over that edge and it chose Italy, one of its major global locations going back to the Roman Empire. An Italian 'Covid' crisis was manufactured in a particular area called Lombardy which just happens to be notorious for its toxic air and therefore respiratory disease. Wuhan, China, déjà vu. An hysterical media told horror stories of Italians dying from 'Covid' in their droves and how Lombardy hospitals were being overrun by a tidal wave of desperately ill people needing treatment after being struck down by the 'deadly virus'. Here was the psychological turning point the Cult had planned. Wow, if this is happening in Italy, the Western mind concluded, this indeed could happen to me and my family. Another point is that Italian authorities responded by following the Chinese blueprint so vehemently recommended by the Cult-owned World Health Organization. They imposed fascistic lockdowns on the whole country viciously policed with the help of surveillance drones sweeping through the streets seeking out anyone who escaped from mass house arrest. Livelihoods were destroyed and psychology unravelled in the way we have witnessed since in all lockdown countries. Crucial to the plan was that Italy responded in this way to set the precedent of suspending freedom and imposing fascism in a 'Western liberal democracy'. I emphasised in an

animated video explanation on davidicke.com posted in the summer of 2020 how important it was to the Cult to expand the Chinese lockdown model across the West. Without this, and the bare-faced lie that non-symptomatic people could still transmit a 'disease' they didn't have, there was no way locking down the whole population, sick and not sick, could be pulled off. At just the right time and with no evidence Cult operatives and gofers claimed that people without symptoms could pass on the 'disease'. In the name of protecting the 'vulnerable' like elderly people, who lockdowns would kill by the tens of thousands, we had for the first time healthy people told to isolate as well as the sick. The great majority of people who tested positive had no symptoms because there was nothing wrong with them. It was just a trick made possible by a test not testing for the 'virus'.

Months after my animated video the Gates-funded Professor Neil Ferguson at the Gates-funded Imperial College confirmed that I was right. He didn't say it in those terms, naturally, but he did say it. Ferguson will enter the story shortly for his outrageously crazy 'computer models' that led to Britain, the United States and many other countries following the Chinese and now Italian methods of response. Put another way, following the Cult script. Ferguson said that SAGE, the UK government's scientific advisory group which has controlled 'Covid' policy from the start, wanted to follow the Chinese lockdown model (while they all continued to work and be paid), but they wondered if they could possibly, in Ferguson's words, 'get away with it in Europe'. 'Get away with it'? Who the hell do these moronic, arrogant people think they are? This appalling man Ferguson said that once Italy went into national lockdown they realised they, too, could mimic China:

> It's a communist one-party state, we said. We couldn't get away with it in Europe, we thought … and then Italy did it. And we realised we could. Behind this garbage from Ferguson is a simple fact: Doing the same as China in every country was the plan from the start and Ferguson's 'models' would play a central role in achieving that. It's just a coincidence, of course, and absolutely nothing to worry your little head about.

Oops, sorry, our mistake

Once the Italian segment of the Psyop had done the job it was designed to do a very different story emerged. Italian authorities revealed that 99 percent of those who had 'died from Covid-19' in Italy had one, two, three, or more 'co-morbidities' or illnesses and health problems that could have ended their life. The US Centers for Disease Control and Prevention (CDC) published a figure of 94 percent for Americans dying of 'Covid' while having other serious medical conditions – on average two to three (some five or six) other potential causes of death. In terms of death from an unproven 'virus' I say it is 100 percent. The other one percent in Italy and six percent in the US would presumably have died from 'Covid's' flu-like symptoms with a range of other possible causes in conjunction with a test not testing for the 'virus'. Fox News reported that even more startling figures had emerged in one US county in which 410 of 422 deaths attributed to 'Covid-19' had other potentially deadly health conditions. The Italian National Health Institute said later that the average age of people dying with a 'Covid-19' diagnosis in Italy was about 81. Ninety percent were over 70 with ten percent over 90. In terms of other reasons to die some 80 percent had two or more chronic diseases with half having three or more including cardiovascular problems, diabetes, respiratory problems and cancer. Why is the phantom 'Covid-19' said to kill overwhelmingly old people and hardly affect the young? Old people continually die of many causes and especially respiratory disease which you can re-diagnose 'Covid-19' while young people die in tiny numbers by comparison and rarely of respiratory disease. Old people 'die of Covid' because they die of other things that can be redesignated 'Covid' and it really is that simple.

Flu has flown

The blueprint was in place. Get your illusory 'cases' from a test not testing for the 'virus' and redesignate other causes of death as 'Covid-19'. You have an instant 'pandemic' from something that is nothing more than a computer-generated fiction. With near-on a

billion people having 'flu-like' symptoms every year the potential was limitless and we can see why flu quickly and apparently miraculously disappeared *worldwide* by being diagnosed 'Covid-19'. The painfully bloody obvious was explained away by the childlike media in headlines like this in the UK '*Independent*': 'Not a single case of flu detected by Public Health England this year as Covid restrictions suppress virus'. I kid you not. The masking, social distancing and house arrest that did not make the 'Covid virus' disappear somehow did so with the 'flu virus'. Even worse the article, by a bloke called Samuel Lovett, suggested that maybe the masking, sanitising and other 'Covid' measures should continue to keep the flu away. With a ridiculousness that disturbs your breathing (it's 'Covid-19') the said Lovett wrote: 'With widespread social distancing and mask-wearing measures in place throughout the UK, the usual routes of transmission for influenza have been blocked.' He had absolutely no evidence to support that statement, but look at the consequences of him acknowledging the obvious. With flu not disappearing at all and only being relabelled 'Covid-19' he would have to contemplate that 'Covid' was a hoax on a scale that is hard to imagine. You need guts and commitment to truth to even go there and that's clearly something Samuel Lovett does not have in abundance. He would never have got it through the editors anyway.

Tens of thousands die in the United States alone every winter from flu including many with pneumonia complications. CDC figures record *45 million* Americans diagnosed with flu in 2017-2018 of which 61,000 died and some reports claim 80,000. Where was the same hysteria then that we have seen with 'Covid-19'? Some 250,000 Americans are admitted to hospital with pneumonia every year with about 50,000 cases proving fatal. About 65 million suffer respiratory disease every year and three million deaths makes this the third biggest cause of death worldwide. You only have to redesignate a portion of all these people 'Covid-19' and you have an instant global pandemic or the *appearance* of one. Why would doctors do this? They are told to do this and all but a few dare not refuse those who must be obeyed. Doctors in general are not researching their own

knowledge and instead take it direct and unquestioned from the authorities that own them and their careers. The authorities say they must now diagnose these symptoms 'Covid-19' and not flu, or whatever, and they do it. Dark suits say put 'Covid-19' on death certificates no matter what the cause of death and the doctors do it. Renegade Minds don't fall for the illusion that doctors and medical staff are all highly-intelligent, highly-principled, seekers of medical truth. *Some are*, but not the majority. They are repeaters, gofers, and yes sir, no sir, purveyors of what the system demands they purvey. The 'Covid' con is not merely confined to diseases of the lungs. Instructions to doctors to put 'Covid-19' on death certificates for anyone dying of *anything* within 28 days (or much more) of a positive test not testing for the 'virus' opened the floodgates. The term dying *with* 'Covid' and not *of* 'Covid' was coined to cover the truth. Whether it was a *with* or an *of* they were all added to the death numbers attributed to the 'deadly virus' compiled by national governments and globally by the Gates-funded Johns Hopkins operation in the United States that was so involved in those 'pandemic' simulations. Fraudulent deaths were added to the ever-growing list of fraudulent 'cases' from false positives from a false test. No wonder Professor Walter Ricciardi, scientific advisor to the Italian minister of health, said after the Lombardy hysteria had done its job that 'Covid' death rates were due to Italy having the second oldest population in the world and to *how hospitals record deaths*:

> The way in which we code deaths in our country is very generous in the sense that all the people who die in hospitals with the coronavirus are deemed to be dying of the coronavirus. On re-evaluation by the National Institute of Health, only 12 per cent of death certificates have shown a direct causality from coronavirus, while 88 per cent of patients who have died have at least one pre-morbidity – many had two or three.

This is extraordinary enough when you consider the propaganda campaign to use Italy to terrify the world, but how can they even say twelve percent were genuine when the 'virus' has not been shown to exist, its 'code' is a computer program, and diagnosis comes from a test not testing for it? As in China, and soon the world, 'Covid-19' in

Italy was a redesignation of diagnosis. Lies and corruption were to become the real 'pandemic' fuelled by a pathetically-compliant medical system taking its orders from the tiny few at the top of their national hierarchy who answered to the World Health Organization which answers to Gates and the Cult. Doctors were told – ordered – to diagnose a particular set of symptoms 'Covid-19' and put that on the death certificate for any cause of death if the patient had tested positive with a test not testing for the virus or had 'Covid' symptoms like the flu. The United States even introduced big financial incentives to manipulate the figures with hospitals receiving £4,600 from the Medicare system for diagnosing someone with regular pneumonia, $13,000 if they made the diagnosis from the same symptoms 'Covid-19' pneumonia, and $39, 000 if they put a 'Covid' diagnosed patient on a ventilator that would almost certainly kill them. A few – painfully and pathetically few – medical whistleblowers revealed (before Cult-owned YouTube deleted their videos) that they had been instructed to 'let the patient crash' and put them straight on a ventilator instead of going through a series of far less intrusive and dangerous methods as they would have done before the pandemic hoax began and the financial incentives kicked in. We are talking cold-blooded murder given that ventilators are so damaging to respiratory systems they are usually the last step before heaven awaits. Renegade Minds never fall for the belief that people in white coats are all angels of mercy and cannot be full-on psychopaths. I have explained in detail in *The Answer* how what I am describing here played out across the world coordinated by the World Health Organization through the medical hierarchies in almost every country.

Medical scientist calls it

Information about the non-existence of the 'virus' began to emerge for me in late March, 2020, and mushroomed after that. I was sent an email by Sir Julian Rose, a writer, researcher, and organic farming promotor, from a medical scientist friend of his in the United States. Even at that early stage in March the scientist was able to explain

how the 'Covid' hoax was being manipulated. He said there were no reliable tests for a specific 'Covid-19 virus' and nor were there any reliable agencies or media outlets for reporting numbers of actual 'Covid-19' cases. We have seen in the long period since then that he was absolutely right. 'Every action and reaction to Covid-19 is based on totally flawed data and we simply cannot make accurate assessments,' he said. Most people diagnosed with 'Covid-19' were showing nothing more than cold and flu-like symptoms 'because most coronavirus strains *are* nothing more than cold/flu-like symptoms'. We had farcical situations like an 84-year-old German man testing positive for 'Covid-19' and his nursing home ordered to quarantine only for him to be found to have a common cold. The scientist described back then why PCR tests and what he called the 'Mickey Mouse test kits' were useless for what they were claimed to be identifying. 'The idea these kits can isolate a specific virus like Covid-19 is nonsense,' he said. Significantly, he pointed out that 'if you want to create a totally false panic about a totally false pandemic – pick a coronavirus'. This is exactly what the Cult-owned Gates, World Economic Forum and Johns Hopkins University did with their Event 201 'simulation' followed by their real-life simulation called the 'pandemic'. The scientist said that all you had to do was select the sickest of people with respiratory-type diseases in a single location – 'say Wuhan' – and administer PCR tests to them. You can then claim that anyone showing 'viral sequences' similar to a coronavirus 'which will inevitably be quite a few' is suffering from a 'new' disease:

> Since you already selected the sickest flu cases a fairly high proportion of your sample will go on to die. You can then say this 'new' virus has a CFR [case fatality rate] higher than the flu and use this to infuse more concern and do more tests which will of course produce more 'cases', which expands the testing, which produces yet more 'cases' and so on and so on. Before long you have your 'pandemic', and all you have done is use a simple test kit trick to convert the worst flu and pneumonia cases into something new that doesn't ACTUALLY EXIST [my emphasis].

He said that you then 'just run the same scam in other countries' and make sure to keep the fear message running high 'so that people

will feel panicky and less able to think critically'. The only problem to overcome was the fact *there is no* actual new deadly pathogen and only regular sick people. This meant that deaths from the 'new deadly pathogen' were going to be way too low for a real new deadly virus pandemic, but he said this could be overcome in the following ways – all of which would go on to happen:

1. You can claim this is just the beginning and more deaths are imminent [you underpin this with fantasy 'computer projections']. Use this as an excuse to quarantine everyone and then claim the quarantine prevented the expected millions of dead.

2. You can [say that people] 'minimizing' the dangers are irresponsible and bully them into not talking about numbers.

3. You can talk crap about made up numbers hoping to blind people with pseudoscience.

4. You can start testing well people (who, of course, will also likely have shreds of coronavirus [RNA] in them) and thus inflate your 'case figures' with 'asymptomatic carriers' (you will of course have to spin that to sound deadly even though any virologist knows the more symptom-less cases you have the less deadly is your pathogen).

The scientist said that if you take these simple steps 'you can have your own entirely manufactured pandemic up and running in weeks'. His analysis made so early in the hoax was brilliantly prophetic of what would actually unfold. Pulling all the information together in these recent chapters we have this is simple 1, 2, 3, of how you can delude virtually the entire human population into believing in a 'virus' that doesn't exist:

- A 'Covid case' is someone who tests positive with a test not testing for the 'virus'.

- A 'Covid death' is someone who dies of *any cause* within 28 days (or much longer) of testing positive with a test not testing for the 'virus.

- Asymptomatic means there is nothing wrong with you, but they claim you can pass on what you don't have to justify locking

down (quarantining) healthy people in totality.

The foundations of the hoax are that simple. A study involving ten million people in Wuhan, published in November, 2020, demolished the whole lie about those without symptoms passing on the 'virus'. They found '300 asymptomatic cases' and traced their contacts to find that not one of them was detected with the 'virus'. 'Asymptomatic' patients and their contacts were isolated for no less than two weeks and nothing changed. I know it's all crap, but if you are going to claim that those without symptoms can transmit 'the virus' then you must produce evidence for that and they never have. Even World Health Organization official Dr Maria Van Kerkhove, head of the emerging diseases and zoonosis unit, said as early as June, 2020, that she doubted the validity of asymptomatic transmission. She said that 'from the data we have, it still seems to be rare that an asymptomatic person actually transmits onward to a secondary individual' and by 'rare' she meant that she couldn't cite any case of asymptomatic transmission.

The Ferguson factor

The problem for the Cult as it headed into March, 2020, when the script had lockdown due to start, was that despite all the manipulation of the case and death figures they still did not have enough people alleged to have died from 'Covid' to justify mass house arrest. This was overcome in the way the scientist described: 'You can claim this is just the beginning and more deaths are imminent … Use this as an excuse to quarantine everyone and then claim the quarantine prevented the expected millions of dead.' Enter one Professor Neil Ferguson, the Gates-funded 'epidemiologist' at the Gates-funded Imperial College in London. Ferguson is Britain's Christian Drosten in that he has a dire record of predicting health outcomes, but is still called upon to advise government on the next health outcome when another 'crisis' comes along. This may seem to be a strange and ridiculous thing to do. Why would you keep turning for policy guidance to people who have a history of being

monumentally wrong? Ah, but it makes sense from the Cult point of view. These 'experts' keep on producing predictions that suit the Cult agenda for societal transformation and so it was with Neil Ferguson as he revealed his horrific (and clearly insane) computer model predictions that allowed lockdowns to be imposed in Britain, the United States and many other countries. Ferguson does not have even an A-level in biology and would appear to have no formal training in computer modelling, medicine or epidemiology, according to Derek Winton, an MSc in Computational Intelligence. He wrote an article somewhat aghast at what Ferguson did which included taking no account of respiratory disease 'seasonality' which means it is far worse in the winter months. Who would have thought that respiratory disease could be worse in the winter? Well, certainly not Ferguson.

The massively China-connected Imperial College and its bizarre professor provided the excuse for the long-incubated Chinese model of human control to travel westward at lightning speed. Imperial College confirms on its website that it collaborates with the Chinese Research Institute; publishes more than 600 research papers every year with Chinese research institutions; has 225 Chinese staff; 2,600 Chinese students – the biggest international group; 7,000 former students living in China which is the largest group outside the UK; and was selected for a tour by China's President Xi Jinping during his state visit to the UK in 2015. The college takes major donations from China and describes itself as the UK's number one university collaborator with Chinese research institutions. The China communist/fascist government did not appear phased by the woeful predictions of Ferguson and Imperial when during the lockdown that Ferguson induced the college signed a five-year collaboration deal with China tech giant Huawei that will have Huawei's indoor 5G network equipment installed at the college's West London tech campus along with an 'AI cloud platform'. The deal includes Chinese sponsorship of Imperial's Venture Catalyst entrepreneurship competition. Imperial is an example of the enormous influence the Chinese government has within British and North American

universities and research centres – and further afield. Up to 200 academics from more than a dozen UK universities are being investigated on suspicion of 'unintentionally' helping the Chinese government build weapons of mass destruction by 'transferring world-leading research in advanced military technology such as aircraft, missile designs and cyberweapons'. Similar scandals have broken in the United States, but it's all a coincidence. Imperial College serves the agenda in many other ways including the promotion of every aspect of the United Nations Agenda 21/2030 (the Great Reset) and produced computer models to show that human-caused 'climate change' is happening when in the real world it isn't. Imperial College is driving the climate agenda as it drives the 'Covid' agenda (both Cult hoaxes) while Patrick Vallance, the UK government's Chief Scientific Adviser on 'Covid', was named Chief Scientific Adviser to the UN 'climate change' conference known as COP26 hosted by the government in Glasgow, Scotland. 'Covid' and 'climate' are fundamentally connected.

Professor Woeful

From Imperial's bosom came Neil Ferguson still advising government despite his previous disasters and it was announced early on that he and other key people like UK Chief Medical Adviser Chris Whitty had caught the 'virus' as the propaganda story was being sold. Somehow they managed to survive and we had Prime Minister Boris Johnson admitted to hospital with what was said to be a severe version of the 'virus' in this same period. His whole policy and demeanour changed when he returned to Downing Street. It's a small world with these government advisors – especially in their communal connections to Gates – and Ferguson had partnered with Whitty to write a paper called 'Infectious disease: Tough choices to reduce Ebola transmission' which involved another scare-story that didn't happen. Ferguson's 'models' predicted that up to150, 000 could die from 'mad cow disease', or BSE, and its version in sheep if it was transmitted to humans. BSE was not transmitted and instead triggered by an organophosphate pesticide used to treat a pest on

cows. Fewer than 200 deaths followed from the human form. Models by Ferguson and his fellow incompetents led to the unnecessary culling of millions of pigs, cattle and sheep in the foot and mouth outbreak in 2001 which destroyed the lives and livelihoods of farmers and their families who had often spent decades building their herds and flocks. Vast numbers of these animals did not have foot and mouth and had no contact with the infection. Another 'expert' behind the cull was Professor Roy Anderson, a computer modeller at Imperial College specialising in the epidemiology of *human*, not animal, disease. Anderson has served on the Bill and Melinda Gates Grand Challenges in Global Health advisory board and chairs another Gates-funded organisation. Gates is everywhere.

In a precursor to the 'Covid' script Ferguson backed closing schools 'for prolonged periods' over the swine flu 'pandemic' in 2009 and said it would affect a third of the world population if it continued to spread at the speed he claimed to be happening. His mates at Imperial College said much the same and a news report said: 'One of the authors, the epidemiologist and disease modeller Neil Ferguson, who sits on the World Health Organisation's emergency committee for the outbreak, said the virus had "full pandemic potential".' Professor Liam Donaldson, the Chris Whitty of his day as Chief Medical Officer, said the worst case could see 30 percent of the British people infected by swine flu with 65,000 dying. Ferguson and Donaldson were indeed proved correct when at the end of the year the number of deaths attributed to swine flu was 392. The term 'expert' is rather liberally applied unfortunately, not least to complete idiots. Swine flu 'projections' were great for GlaxoSmithKline (GSK) as millions rolled in for its Pandemrix influenza vaccine which led to brain damage with children most affected. The British government (taxpayers) paid out more than £60 million in compensation after GSK was given immunity from prosecution. Yet another 'Covid' déjà vu. Swine flu was supposed to have broken out in Mexico, but Dr Wolfgang Wodarg, a German doctor, former member of parliament and critic of the 'Covid' hoax, observed 'the spread of swine flu' in Mexico City at the time. He

said: 'What we experienced in Mexico City was a very mild flu which did not kill more than usual – which killed even fewer people than usual.' Hyping the fear against all the facts is not unique to 'Covid' and has happened many times before. Ferguson is reported to have over-estimated the projected death toll of bird flu (H5N1) by some three million-fold, but bird flu vaccine makers again made a killing from the scare. This is some of the background to the Neil Ferguson who produced the perfectly-timed computer models in early 2020 predicting that half a million people would die in Britain without draconian lockdown and 2.2 million in the United States. Politicians panicked, people panicked, and lockdowns of alleged short duration were instigated to 'flatten the curve' of cases gleaned from a test not testing for the 'virus'. I said at the time that the public could forget the 'short duration' bit. This was an agenda to destroy the livelihoods of the population and force them into mass control through dependency and there was going to be nothing 'short' about it. American researcher Daniel Horowitz described the consequences of the 'models' spewed out by Gates-funded Ferguson and Imperial College:

> What led our government and the governments of many other countries into panic was a single Imperial College of UK study, funded by global warming activists, that predicted 2.2 million deaths if we didn't lock down the country. In addition, the reported 8-9% death rate in Italy scared us into thinking there was some other mutation of this virus that they got, which might have come here.
>
> Together with the fact that we were finally testing and had the ability to actually report new cases, we thought we were headed for a death spiral. But again … we can't flatten a curve if we don't know when the curve started.

How about it *never* started?

Giving them what they want

An investigation by German news outlet *Welt Am Sonntag (World on Sunday)* revealed how in March, 2020, the German government gathered together 'leading scientists from several research institutes and universities' and 'together, they were to produce a [modelling]

paper that would serve as legitimization for further tough political measures'. The Cult agenda was justified by computer modelling not based on evidence or reality; it was specifically constructed to justify the Cult demand for lockdowns all over the world to destroy the independent livelihoods of the global population. All these modellers and everyone responsible for the 'Covid' hoax have a date with a trial like those in Nuremberg after World War Two when Nazis faced the consequences of their war crimes. These corrupt-beyond-belief 'modellers' wrote the paper according to government instructions and it said that that if lockdown measures were lifted then up to one million Germans would die from 'Covid-19' adding that some would die 'agonizingly at home, gasping for breath' unable to be treated by hospitals that couldn't cope. All lies. No matter – it gave the Cult all that it wanted. What did long-time government 'modeller' Neil Ferguson say? If the UK and the United States didn't lockdown half a million would die in Britain and 2.2 million Americans. Anyone see a theme here? 'Modellers' are such a crucial part of the lockdown strategy that we should look into their background and follow the money. Researcher Rosemary Frei produced an excellent article headlined 'The Modelling-paper Mafiosi'. She highlights a guy called John Edmunds, a British epidemiologist, and professor in the Faculty of Epidemiology and Population Health at the London School of Hygiene & Tropical Medicine. He studied at Imperial College. Edmunds is a member of government 'Covid' advisory bodies which have been dictating policy, the New and Emerging Respiratory Virus Threats Advisory Group (NERVTAG) and the Scientific Advisory Group for Emergencies (SAGE).

Ferguson, another member of NERVTAG and SAGE, led the way with the original 'virus' and Edmunds has followed in the 'variant' stage and especially the so-called UK or Kent variant known as the 'Variant of Concern' (VOC) B.1.1.7. He said in a co-written report for the Centre for Mathematical modelling of Infectious Diseases at the London School of Hygiene and Tropical Medicine, with input from the Centre's 'Covid-19' Working Group, that there was 'a realistic

possibility that VOC B.1.1.7 is associated with an increased risk of death compared to non-VOC viruses'. Fear, fear, fear, get the vaccine, fear, fear, fear, get the vaccine. Rosemary Frei reveals that almost all the paper's authors and members of the modelling centre's 'Covid-19' Working Group receive funding from the Bill and Melinda Gates Foundation and/or the associated Gates-funded Wellcome Trust. The paper was published by e-journal *Medr χiv* which only publishes papers not peer-reviewed and the journal was established by an organisation headed by Facebook's Mark Zuckerberg and his missus. What a small world it is. Frei discovered that Edmunds is on the Scientific Advisory Board of the Coalition for Epidemic Preparedness Innovations (CEPI) which was established by the Bill and Melinda Gates Foundation, Klaus Schwab's Davos World Economic Forum and Big Pharma giant Wellcome. CEPI was 'launched in Davos [in 2017] to develop vaccines to stop future epidemics', according to its website. 'Our mission is to accelerate the development of vaccines against emerging infectious diseases and enable equitable access to these vaccines for people during outbreaks.' What kind people they are. Rosemary Frei reveals that Public Health England (PHE) director Susan Hopkins is an author of her organisation's non-peer-reviewed reports on 'new variants'. Hopkins is a professor of infectious diseases at London's Imperial College which is gifted tens of millions of dollars a year by the Bill and Melinda Gates Foundation. Gates-funded modelling disaster Neil Ferguson also co-authors Public Health England reports and he spoke in December, 2020, about the potential danger of the B.1.1.7. 'UK variant' promoted by Gates-funded modeller John Edmunds. When I come to the 'Covid vaccines' the 'new variants' will be shown for what they are – bollocks.

Connections, connections

All these people and modellers are lockdown-obsessed or, put another way, they demand what the Cult demands. Edmunds said in January, 2021, that to ease lockdowns too soon would be a disaster and they had to 'vaccinate much, much, much more widely than the

elderly'. Rosemary Frei highlights that Edmunds is married to Jeanne Pimenta who is described in a LinkedIn profile as director of epidemiology at GlaxoSmithKline (GSK) and she held shares in the company. Patrick Vallance, co-chair of SAGE and the government's Chief Scientific Adviser, is a former executive of GSK and has a deferred bonus of shares in the company worth £600,000. GSK has serious business connections with Bill Gates and is collaborating with mRNA-'vaccine' company CureVac to make 'vaccines' for the new variants that Edmunds is talking about. GSK is planning a 'Covid vaccine' with drug giant Sanofi. Puppet Prime Minister Boris Johnson announced in the spring of 2021 that up to 60 million vaccine doses were to be made at the GSK facility at Barnard Castle in the English North East. Barnard Castle, with a population of just 6,000, was famously visited in breach of lockdown rules in April, 2020, by Johnson aide Dominic Cummings who said that he drove there 'to test his eyesight' before driving back to London. Cummings would be better advised to test his integrity – not that it would take long. The GSK facility had nothing to do with his visit then although I'm sure Patrick Vallance would have been happy to arrange an introduction and some tea and biscuits. Ruthless psychopath Gates has made yet another fortune from vaccines in collaboration with Big Pharma companies and gushes at the phenomenal profits to be made from vaccines – more than a 20-to-1 return as he told one interviewer. Gates also tweeted in December, 2019, with the foreknowledge of what was coming: 'What's next for our foundation? I'm particularly excited about what the next year could mean for one of the best buys in global health: vaccines.'

Modeller John Edmunds is a big promotor of vaccines as all these people appear to be. He's the dean of the London School of Hygiene & Tropical Medicine's Faculty of Epidemiology and Population Health which is primarily funded by the Bill and Melinda Gates Foundation and the Gates-established and funded GAVI vaccine alliance which is the Gates vehicle to vaccinate the world. The organisation Doctors Without Borders has described GAVI as being 'aimed more at supporting drug-industry desires to promote new

products than at finding the most efficient and sustainable means for fighting the diseases of poverty'. But then that's why the psychopath Gates created it. John Edmunds said in a video that the London School of Hygiene & Tropical Medicine is involved in every aspect of vaccine development including large-scale clinical trials. He contends that mathematical modelling can show that vaccines protect individuals and society. That's on the basis of shit in and shit out, I take it. Edmunds serves on the UK Vaccine Network as does Ferguson and the government's foremost 'Covid' adviser, the grim-faced, dark-eyed Chris Whitty. The Vaccine Network says it works 'to support the government to identify and shortlist targeted investment opportunities for the most promising vaccines and vaccine technologies that will help combat infectious diseases with epidemic potential, and to address structural issues related to the UK's broader vaccine infrastructure'. Ferguson is acting Director of the Imperial College Vaccine Impact Modelling Consortium which has funding from the Bill and Melina Gates Foundation and the Gates-created GAVI 'vaccine alliance'. Anyone wonder why these characters see vaccines as the answer to every problem? Ferguson is wildly enthusiastic in his support for GAVI's campaign to vaccine children en masse in poor countries. You would expect someone like Gates who has constantly talked about the need to reduce the population to want to fund vaccines to keep more people alive. I'm sure that's why he does it. The John Edmunds London School of Hygiene & Tropical Medicine (LSHTM) has a Vaccines Manufacturing Innovation Centre which develops, tests and commercialises vaccines. Rosemary Frei writes:

> The vaccines centre also performs affiliated activities like combating 'vaccine hesitancy'. The latter includes the Vaccine Confidence Project. The project's stated purpose is, among other things, 'to provide analysis and guidance for early response and engagement with the public to ensure sustained confidence in vaccines and immunisation'. The Vaccine Confidence Project's director is LSHTM professor Heidi Larson. For more than a decade she's been researching how to combat vaccine hesitancy.

How the bloody hell can blokes like John Edmunds and Neil Ferguson with those connections and financial ties model 'virus' case

and death projections for the government and especially in a way that gives their paymasters like Gates exactly what they want? It's insane, but this is what you find throughout the world.

'Covid' is not dangerous, oops, wait, yes it is

Only days before Ferguson's nightmare scenario made Jackboot Johnson take Britain into a China-style lockdown to save us from a deadly 'virus' the UK government website gov.uk was reporting something very different to Ferguson on a page of official government guidance for 'high consequence infectious diseases (HCID)'. It said this about 'Covid-19':

> As of 19 March 2020, COVID-19 *is no longer considered to be a high consequence infectious diseases (HCID) in the UK* [my emphasis]. The 4 nations public health HCID group made an interim recommendation in January 2020 to classify COVID-19 as an HCID. This was based on consideration of the UK HCID criteria about the virus and the disease with information available during the early stages of the outbreak.
>
> Now that more is known about COVID-19, the public health bodies in the UK have reviewed the most up to date information about COVID-19 against the UK HCID criteria. They have determined that several features have now changed; in particular, more information is available about mortality rates (low overall), and there is now greater clinical awareness and a specific and sensitive laboratory test, the availability of which continues to increase. The Advisory Committee on Dangerous Pathogens (ACDP) is also of the opinion that COVID-19 should no longer be classified as an HCID.

Soon after the government had been exposed for downgrading the risk they upgraded it again and everyone was back to singing from the same Cult hymn book. Ferguson and his fellow Gates clones indicated that lockdowns and restrictions would have to continue until a Gates-funded vaccine was developed. Gates said the same because Ferguson and his like were repeating the Gates script which is the Cult script. 'Flatten the curve' became an ongoing nightmare of continuing lockdowns with periods in between of severe restrictions in pursuit of destroying independent incomes and had nothing to do with protecting health about which the Cult gives not a shit. Why wouldn't Ferguson be pushing a vaccine 'solution' when he's owned by vaccine-obsessive Gates who makes a fortune from them and

when Ferguson heads the Vaccine Impact Modelling Consortium at Imperial College funded by the Gates Foundation and GAVI, the 'vaccine alliance', created by Gates as his personal vaccine promotion operation? To compound the human catastrophe that Ferguson's 'models' did so much to create he was later exposed for breaking his own lockdown rules by having sexual liaisons with his married girlfriend Antonia Staats at his home while she was living at another location with her husband and children. Staats was a 'climate' activist and senior campaigner at the Soros-funded Avaaz which I wouldn't trust to tell me that grass is green. Ferguson had to resign as a government advisor over this hypocrisy in May, 2020, but after a period of quiet he was back being quoted by the ridiculous media on the need for more lockdowns and a vaccine rollout. Other government-advising 'scientists' from Imperial College' held the fort in his absence and said lockdown could be indefinite until a vaccine was found. The Cult script was being sung by the payrolled choir. I said there was no intention of going back to 'normal' when the 'vaccine' came because the 'vaccine' is part of a very different agenda that I will discuss in Human 2.0. Why would the Cult want to let the world go back to normal when destroying that normal forever was the whole point of what was happening? House arrest, closing businesses and schools through lockdown, (un)social distancing and masks all followed the Ferguson fantasy models. Again as I predicted (these people are so predictable) when the 'vaccine' arrived we were told that house arrest, lockdown, (un)social distancing and masks would still have to continue. I will deal with the masks in the next chapter because they are of fundamental importance.

Where's the 'pandemic'?

Any mildly in-depth assessment of the figures revealed what was really going on. Cult-funded and controlled organisations still have genuine people working within them such is the number involved. So it is with Genevieve Briand, assistant program director of the Applied Economics master's degree program at Johns Hopkins

University. She analysed the impact that 'Covid-19' had on deaths from *all* causes in the United States using official data from the CDC for the period from early February to early September, 2020. She found that allegedly 'Covid' *related*-deaths exceeded those from heart disease which she found strange with heart disease always the biggest cause of fatalities. Her research became even more significant when she noted the sudden decline in 2020 of *all* non-'Covid' deaths: 'This trend is completely contrary to the pattern observed in all previous years … the total decrease in deaths by other causes almost exactly equals the increase in deaths by Covid-19.' This was such a game, set and match in terms of what was happening that Johns Hopkins University deleted the article on the grounds that it 'was being used to support false and dangerous inaccuracies about the impact of the pandemic'. No – because it exposed the scam from official CDC figures and this was confirmed when those figures were published in January, 2021. Here we can see the effect of people dying from heart attacks, cancer, road accidents and gunshot wounds – *anything* – having 'Covid-19' on the death certificate along with those diagnosed from 'symptoms' who had even not tested positive with a test not testing for the 'virus'. I am not kidding with the gunshot wounds, by the way. Brenda Bock, coroner in Grand County, Colorado, revealed that two gunshot victims tested positive for the 'virus' within the previous 30 days and were therefore classified as 'Covid deaths'. Bock said: 'These two people had tested positive for Covid, but that's not what killed them. A gunshot wound is what killed them.' She said she had not even finished her investigation when the state listed the gunshot victims as deaths due to the 'virus'. The death and case figures for 'Covid-19' are an absolute joke and yet they are repeated like parrots by the media, politicians and alleged medical 'experts'. The official Cult narrative is the only show in town.

Genevieve Briand found that deaths from all causes were not exceptional in 2020 compared with previous years and a Spanish magazine published figures that said the same about Spain which was a 'Covid' propaganda hotspot at one point. *Discovery Salud*, a

health and medicine magazine, quoted government figures which showed how 17,000 *fewer* people died in Spain in 2020 than in 2019 and more than 26,000 fewer than in 2018. The age-standardised mortality rate for England and Wales when age distribution is taken into account was significantly lower in 2020 than the 1970s, 80s and 90s, and was only the ninth highest since 2000. Where is the 'pandemic'?

Post mortems and autopsies virtually disappeared for 'Covid' deaths amid claims that 'virus-infected' bodily fluids posed a risk to those carrying out the autopsy. This was rejected by renowned German pathologist and forensic doctor Klaus Püschel who said that he and his staff had by then done 150 autopsies on 'Covid' patients with no problems at all. He said they were needed to know why some 'Covid' patients suffered blood clots and not severe respiratory infections. The 'virus' is, after all, called SARS or 'severe acute respiratory syndrome'. I highlighted in the spring of 2020 this phenomenon and quoted New York intensive care doctor Cameron Kyle-Sidell who posted a soon deleted YouTube video to say that they had been told to prepare to treat an infectious disease called 'Covid-19', but that was not what they were dealing with. Instead he likened the lung condition of the most severely ill patients to what you would expect with cabin depressurisation in a plane at 30,000 feet or someone dropped on the top of Everest without oxygen or acclimatisation. I have never said this is not happening to a small minority of alleged 'Covid' patients – I am saying this is not caused by a phantom 'contagious virus'. Indeed Kyle-Sidell said that 'Covid-19' was not the disease they were told was coming their way. 'We are operating under a medical paradigm that is untrue,' he said, and he believed they were treating the wrong disease: 'These people are being slowly starved of oxygen.' Patients would take off their oxygen masks in a state of fear and stress and while they were blue in the face on the brink of death. They did not look like patients dying of pneumonia. You can see why they don't want autopsies when their virus doesn't exist and there is another condition in some people that they don't wish to be uncovered. I should add here that

the 5G system of millimetre waves was being rapidly introduced around the world in 2020 and even more so now as they fire 5G at the Earth from satellites. At 60 gigahertz within the 5G range that frequency interacts with the oxygen molecule and stops people breathing in sufficient oxygen to be absorbed into the bloodstream. They are installing 5G in schools and hospitals. The world is not mad or anything. 5G can cause major changes to the lungs and blood as I detail in *The Answer* and these consequences are labelled 'Covid-19', the alleged symptoms of which can be caused by 5G and other electromagnetic frequencies as cells respond to radiation poisoning.

The 'Covid death' scam

Dr Scott Jensen, a Minnesota state senator and medical doctor, exposed 'Covid' Medicare payment incentives to hospitals and death certificate manipulation. He said he was sent a seven-page document by the US Department of Health 'coaching' him on how to fill out death certificates which had never happened before. The document said that he didn't need to have a laboratory test for 'Covid-19' to put that on the death certificate and that shocked him when death certificates are supposed to be about facts. Jensen described how doctors had been 'encouraged, if not pressured' to make a diagnosis of 'Covid-19' if they thought it was probable or *'presumed*'. No positive test was necessary – not that this would have mattered anyway. He said doctors were told to diagnose 'Covid' by symptoms when these were the same as colds, allergies, other respiratory problems, and certainly with influenza which 'disappeared' in the 'Covid' era. A common sniffle was enough to get the dreaded verdict. Ontario authorities decreed that a single care home resident with *one* symptom from a long list must lead to the isolation of the entire home. Other courageous doctors like Jensen made the same point about death figure manipulation and how deaths by other causes were falling while 'Covid-19 deaths' were rising at the same rate due to re-diagnosis. Their videos rarely survive long on YouTube with its Cult-supporting algorithms courtesy of CEO Susan Wojcicki and her bosses at Google. Figure-tampering was so glaring

and ubiquitous that even officials were letting it slip or outright saying it. UK chief scientific adviser Patrick Vallance said on one occasion that 'Covid' on the death certificate doesn't mean 'Covid' was the cause of death (so why the hell is it there?) and we had the rare sight of a BBC reporter telling the truth when she said: 'Someone could be successfully treated for Covid, in say April, discharged, and then in June, get run over by a bus and die … That person would still be counted as a Covid death in England.' Yet the BBC and the rest of the world media went on repeating the case and death figures as if they were real. Illinois Public Health Director Dr Ngozi Ezike revealed the deceit while her bosses must have been clenching their buttocks:

> If you were in a hospice and given a few weeks to live and you were then found to have Covid that would be counted as a Covid death. [There might be] a clear alternate cause, but it is still listed as a Covid death. So everyone listed as a Covid death doesn't mean that was the cause of the death, but that they had Covid at the time of death.

Yes, a 'Covid virus' never shown to exist and tested for with a test not testing for the 'virus'. In the first period of the pandemic hoax through the spring of 2020 the process began of designating almost everything a 'Covid' death and this has continued ever since. I sat in a restaurant one night listening to a loud conversation on the next table where a family was discussing in bewilderment how a relative who had no symptoms of 'Covid', and had died of a long-term problem, could have been diagnosed a death by the 'virus'. I could understand their bewilderment. If they read this book they will know why this medical fraud has been perpetrated the world over.

Some media truth shock

The media ignored the evidence of death certificate fraud until eventually one columnist did speak out when she saw it first-hand. Bel Mooney is a long-time national newspaper journalist in Britain currently working for the *Daily Mail*. Her article on February 19th, 2021, carried this headline: 'My dad Ted passed three Covid tests

and died of a chronic illness yet he's officially one of Britain's 120,000 victims of the virus and is far from alone ... so how many more are there?' She told how her 99-year-old father was in a care home with a long-standing chronic obstructive pulmonary disease and vascular dementia. Maybe, but he was still aware enough to tell her from the start that there was no 'virus' and he refused the 'vaccine' for that reason. His death was not unexpected given his chronic health problems and Mooney said she was shocked to find that 'Covid-19' was declared the cause of death on his death certificate. She said this was a 'bizarre and unacceptable untruth' for a man with long-time health problems who had tested negative twice at the home for the 'virus'. I was also shocked by this story although not by what she said. I had been highlighting the death certificate manipulation for ten months. It was the confirmation that a professional full-time journalist only realised this was going on when it affected her directly and neither did she know that whether her dad tested positive or negative was irrelevant with the test not testing for the 'virus'. Where had she been? She said she did not believe in 'conspiracy theories' without knowing I'm sure that this and 'conspiracy theorists' were terms put into widespread circulation by the CIA in the 1960s to discredit those who did not accept the ridiculous official story of the Kennedy assassination. A blanket statement of 'I don't believe in conspiracy theories' is always bizarre. The dictionary definition of the term alone means the world is drowning in conspiracies. What she said was even more daft when her dad had just been affected by the 'Covid' conspiracy. Why else does she think that 'Covid-19' was going on the death certificates of people who died of something else?

To be fair once she saw from personal experience what was happening she didn't mince words. Mooney was called by the care home on the morning of February 9th to be told her father had died in his sleep. When she asked for the official cause of death what came back was 'Covid-19'. Mooney challenged this and was told there had been deaths from Covid on the dementia floor (confirmed by a test not testing for the 'virus') so they considered it 'reasonable

to assume'. 'But doctor,' Mooney rightly protested, 'an assumption isn't a diagnosis.' She said she didn't blame the perfectly decent and sympathetic doctor – 'he was just doing his job'. Sorry, but that's *bullshit*. He wasn't doing his job at all. He was putting a false cause of death on the death certificate and that is a criminal offence for which he should be brought to account and the same with the millions of doctors worldwide who have done the same. They were not doing their job they were following orders and that must not wash at new Nuremberg trials any more than it did at the first ones. Mooney's doctor was 'assuming' (presuming) as he was told to, but 'just following orders' makes no difference to his actions. A doctor's job is to serve the patient and the truth, not follow orders, but that's what they have done all over the world and played a central part in making the 'Covid' hoax possible with all its catastrophic consequences for humanity. Shame on them and they must answer for their actions. Mooney said her disquiet worsened when she registered her father's death by telephone and was told by the registrar there had been very many other cases like hers where 'the deceased' had not tested positive for 'Covid' yet it was recorded as the cause of death. The test may not matter, but those involved at their level *think* it matters and it shows a callous disregard for accurate diagnosis. The pressure to do this is coming from the top of the national 'health' pyramids which in turn obey the World Health Organization which obeys Gates and the Cult. Mooney said the registrar agreed that this must distort the national figures adding that 'the strangest thing is that every winter we record countless deaths from flu, and this winter there have been none. Not one!' She asked if the registrar thought deaths from flu were being misdiagnosed and lumped together with 'Covid' deaths. The answer was a 'puzzled yes'. Mooney said that the funeral director said the same about 'Covid' deaths which had nothing to do with 'Covid'. They had lost count of the number of families upset by this and other funeral companies in different countries have had the same experience. Mooney wrote:

The nightly shroud-waving and shocking close-ups of pain imposed on us by the TV news bewildered and terrified the population into eager compliance with lockdowns. We were invited to 'save the NHS' and to grieve for strangers – the real-life loved ones behind those shocking death counts. Why would the public imagine what I now fear, namely that the way Covid-19 death statistics are compiled might make the numbers seem greater than they are?

Oh, just a little bit – like 100 percent.

Do the maths

Mooney asked why a country would wish to skew its mortality figures by wrongly certifying deaths? What had been going on? Well, if you don't believe in conspiracies you will never find the answer which is that *it's a conspiracy*. She did, however, describe what she had discovered as a 'national scandal'. In reality it's a global scandal and happening everywhere. Pillars of this conspiracy were all put into place before the button was pressed with the Drosten PCR protocol and high amplifications to produce the cases and death certificate changes to secure illusory 'Covid' deaths. Mooney notes that normally two doctors were needed to certify a death, with one having to know the patient, and how the rules were changed in the spring of 2020 to allow one doctor to do this. In the same period 'Covid deaths' were decreed to be all cases where Covid-19 was put on the death certificate even without a positive test or any symptoms. Mooney asked: 'How many of the 30,851 (as of January 15) care home resident deaths with Covid-19 on the certificate (32.4 per cent of all deaths so far) were based on an assumption, like that of my father? And what has that done to our national psyche?'All of them is the answer to the first question and it has devastated and dismantled the national psyche, actually the global psyche, on a colossal scale. In the UK case and death data is compiled by organisations like Public Health England (PHE) and the Office for National Statistics (ONS). Mooney highlights the insane policy of counting a death from any cause as 'Covid-19' if this happens within 28 days of a positive test (with a test not testing for the 'virus') and she points out that ONS statistics reflect deaths 'involving Covid' 'or due to Covid' which meant in practice any

death where 'Covid-19' was mentioned on the death certificate. She described the consequences of this fraud:

> Most people will accept the narrative they are fed, so panicky governments here and in Europe witnessed the harsh measures enacted in totalitarian China and jumped into lockdown. Headlines about Covid deaths tolled like the knell that would bring doomsday to us all. Fear stalked our empty streets. Politicians parroted the frankly ridiculous aim of 'zero Covid' and shut down the economy, while most British people agreed that lockdown was essential and (astonishingly to me, as a patriotic Brit) even wanted more restrictions.
>
> For what? Lies on death certificates? Never mind the grim toll of lives ruined, suicides, schools closed, rising inequality, depression, cancelled hospital treatments, cancer patients in a torture of waiting, poverty, economic devastation, loneliness, families kept apart, and so on. How many lives have been lost as a direct result of lockdown?

She said that we could join in a national chorus of shock and horror at reaching the 120,000 death toll which was surely certain to have been totally skewed all along, but what about the human cost of lockdown justified by these 'death figures'? *The British Medical Journal* had reported a 1,493 percent increase in cases of children taken to Great Ormond Street Hospital with abusive head injuries alone and then there was the effect on families:

> Perhaps the most shocking thing about all this is that families have been kept apart – and obeyed the most irrational, changing rules at the whim of government – because they believed in the statistics. They succumbed to fear, which his generation rejected in that war fought for freedom. Dad (God rest his soul) would be angry. And so am I.

Another theme to watch is that in the winter months when there are more deaths from all causes they focus on 'Covid' deaths and in the summer when the British Lung Foundation says respiratory disease plummets by 80 percent they rage on about 'cases'. Either way fascism on population is always the answer.

Nazi eugenics in the 21st century

Elderly people in care homes have been isolated from their families month after lonely month with no contact with relatives and grandchildren who were banned from seeing them. We were told

that lockdown fascism was to 'protect the vulnerable' like elderly people. At the same time Do Not Resuscitate (DNR) orders were placed on their medical files so that if they needed resuscitation it wasn't done and 'Covid-19' went on their death certificates. Old people were not being 'protected' they were being culled – murdered in truth. DNR orders were being decreed for disabled and young people with learning difficulties or psychological problems. The UK Care Quality Commission, a non-departmental body of the Department of Health and Social Care, found that 34 percent of those working in health and social care were pressured into placing 'do not attempt cardiopulmonary resuscitation' orders on 'Covid' patients who suffered from disabilities and learning difficulties without involving the patient or their families in the decision. UK judges ruled that an elderly woman with dementia should have the DNA-manipulating 'Covid vaccine' against her son's wishes and that a man with severe learning difficulties should have the jab despite his family's objections. Never mind that many had already died. The judiciary always supports doctors and government in fascist dictatorships. They wouldn't dare do otherwise. A horrific video was posted showing fascist officers from Los Angeles police forcibly giving the 'Covid' shot to women with special needs who were screaming that they didn't want it. The same fascists are seen giving the jab to a sleeping elderly woman in a care home. This is straight out of the Nazi playbook. Hitler's Nazis committed mass murder of the mentally ill and physically disabled throughout Germany and occupied territories in the programme that became known as Aktion T4, or just T4. Sabbatian-controlled Hitler and his grotesque crazies set out to kill those they considered useless and unnecessary. The Reich Committee for the Scientific Registering of Hereditary and Congenital Illnesses registered the births of babies identified by physicians to have 'defects'. By 1941 alone more than 5,000 children were murdered by the state and it is estimated that in total the number of innocent people killed in Aktion T4 was between 275,000 and 300,000. Parents were told their children had been sent away for 'special treatment' never to return. It is rather pathetic to see claims about plans for new extermination camps being dismissed today

when the same force behind current events did precisely that 80 years ago. Margaret Sanger was a Cult operative who used 'birth control' to sanitise her programme of eugenics. Organisations she founded became what is now Planned Parenthood. Sanger proposed that 'the whole dysgenic population would have its choice of segregation or sterilization'. These included epileptics, 'feeble-minded', and prostitutes. Sanger opposed charity because it perpetuated 'human waste'. She reveals the Cult mentality and if anyone thinks that extermination camps are a 'conspiracy theory' their naivety is touching if breathtakingly stupid.

If you don't believe that doctors can act with callous disregard for their patients it is worth considering that doctors and medical staff agreed to put government-decreed DNR orders on medical files and do nothing when resuscitation is called for. I don't know what you call such people in your house. In mine they are Nazis from the Josef Mengele School of Medicine. Phenomenal numbers of old people have died worldwide from the effects of lockdown, depression, lack of treatment, the 'vaccine' (more later) and losing the will to live. A common response at the start of the manufactured pandemic was to remove old people from hospital beds and transfer them to nursing homes. The decision would result in a mass cull of elderly people in those homes through lack of treatment – *not* 'Covid'. Care home whistleblowers have told how once the 'Covid' era began doctors would not come to their homes to treat patients and they were begging for drugs like antibiotics that often never came. The most infamous example was ordered by New York governor Andrew Cuomo, brother of a moronic CNN host, who amazingly was given an Emmy Award for his handling of the 'Covid crisis' by the ridiculous Wokers that hand them out. Just how ridiculous could be seen in February, 2021, when a Department of Justice and FBI investigation began into how thousands of old people in New York died in nursing homes after being discharged from hospital to make way for 'Covid' patients on Cuomo's say-so – and how he and his staff covered up these facts. This couldn't have happened to a nicer psychopath. Even then there was a 'Covid' spin. Reports said that

thousands of old people who tested positive for 'Covid' in hospital were transferred to nursing homes to both die of 'Covid' and transmit it to others. No – they were in hospital because they were ill and the fact that they tested positive with a test not testing for the 'virus' is irrelevant. They were ill often with respiratory diseases ubiquitous in old people near the end of their lives. Their transfer out of hospital meant that their treatment stopped and many would go on to die.

They're old. Who gives a damn?

I have exposed in the books for decades the Cult plan to cull the world's old people and even to introduce at some point what they call a 'demise pill' which at a certain age everyone would take and be out of here by law. In March, 2021, Spain legalised euthanasia and assisted suicide following the Netherlands, Belgium, Luxembourg and Canada on the Tiptoe to the demise pill. Treatment of old people by many 'care' homes has been a disgrace in the 'Covid' era. There are many, many, caring staff – I know some. There have, however, been legions of stories about callous treatment of old people and their families. Police were called when families came to take their loved ones home in the light of isolation that was killing them. They became prisoners of the state. Care home residents in insane, fascist Ontario, Canada, were not allowed to leave their *room* once the 'Covid' hoax began. UK staff have even wheeled elderly people away from windows where family members were talking with them. Oriana Criscuolo from Stockport in the English North West dropped off some things for her 80-year-old father who has Parkinson's disease and dementia and she wanted to wave to him through a ground-floor window. She was told that was 'illegal'. When she went anyway they closed the curtains in the middle of the day. Oriana said:

> It's just unbelievable. I cannot understand how care home staff – people who are being paid to care – have become so uncaring. Their behaviour is inhumane and cruel. It's beyond belief.

She was right and this was not a one-off. What a way to end your life in such loveless circumstances. UK registered nurse Nicky Millen, a proper old school nurse for 40 years, said that when she started her career care was based on dignity, choice, compassion and empathy. Now she said 'the things that are important to me have gone out of the window.' She was appalled that people were dying without their loved ones and saying goodbye on iPads. Nicky described how a distressed 89-year-old lady stroked her face and asked her 'how many paracetamol would it take to finish me off'. Life was no longer worth living while not seeing her family. Nicky said she was humiliated in front of the ward staff and patients for letting the lady stroke her face and giving her a cuddle. Such is the dehumanisation that the 'Covid' hoax has brought to the surface. Nicky worked in care homes where patients told her they were being held prisoner. 'I want to live until I die', one said to her. 'I had a lady in tears because she hadn't seen her great-grandson.' Nicky was compassionate old school meeting psychopathic New Normal. She also said she had worked on a 'Covid' ward with no 'Covid' patients. Jewish writer Shai Held wrote an article in March, 2020, which was headlined 'The Staggering, Heartless Cruelty Toward the Elderly'. What he described was happening from the earliest days of lockdown. He said 'the elderly' were considered a group and not unique individuals (the way of the Woke). Shai Held said:

> Notice how the all-too-familiar rhetoric of dehumanization works: 'The elderly' are bunched together as a faceless mass, all of them considered culprits and thus effectively deserving of the suffering the pandemic will inflict upon them. Lost entirely is the fact that the elderly are individual human beings, each with a distinctive face and voice, each with hopes and dreams, memories and regrets, friendships and marriages, loves lost and loves sustained.

'The elderly' have become another dehumanised group for which anything goes and for many that has resulted in cold disregard for their rights and their life. The distinctive face that Held talks about is designed to be deleted by masks until everyone is part of a faceless mass.

'War-zone' hospitals myth

Again and again medical professionals have told me what was really going on and how hospitals 'overrun like war zones' according to the media were virtually empty. The mantra from medical whistleblowers was please don't use my name or my career is over. Citizen journalists around the world sneaked into hospitals to film evidence exposing the 'war-zone' lie. They really *were* largely empty with closed wards and operating theatres. I met a hospital worker in my town on the Isle of Wight during the first lockdown in 2020 who said the only island hospital had never been so quiet. Lockdown was justified by the psychopaths to stop hospitals being overrun. At the same time that the island hospital was near-empty the military arrived here to provide *extra beds*. It was all propaganda to ramp up the fear to ensure compliance with fascism as were never-used temporary hospitals with thousands of beds known as Nightingales and never-used make-shift mortuaries opened by the criminal UK government. A man who helped to install those extra island beds attributed to the army said they were never used and the hospital was empty. Doctors and nurses 'stood around talking or on their phones, wandering down to us to see what we were doing'. There were no masks or social distancing. He accused the useless local island paper, the *County Press*, of 'pumping the fear as if our hospital was overrun and we only have one so it should have been'. He described ambulances parked up with crews outside in deck chairs. When his brother called an ambulance he was told there was a two-hour backlog which he called 'bullshit'. An old lady on the island fell 'and was in a bad way', but a caller who rang for an ambulance was told the situation wasn't urgent enough. Ambulance stations were working under capacity while people would hear ambulances with sirens blaring driving through the streets. When those living near the stations realised what was going on they would follow them as they left, circulated around an urban area with the sirens going, and then came back without stopping. All this was to increase levels of fear and the same goes for the 'ventilator shortage crisis' that cost tens of millions for hastily produced ventilators never to be used.

Ambulance crews that agreed to be exploited in this way for fear propaganda might find themselves a mirror. I wish them well with that. Empty hospitals were the obvious consequence of treatment and diagnoses of non-'Covid' conditions cancelled and those involved handed a death sentence. People have been dying at home from undiagnosed and untreated cancer, heart disease and other life-threatening conditions to allow empty hospitals to deal with a 'pandemic' that wasn't happening.

Death of the innocent

'War-zones' have been laying off nursing staff, even doctors where they can. There was no work for them. Lockdown was justified by saving lives and protecting the vulnerable they were actually killing with DNR orders and preventing empty hospitals being 'overrun'. In Britain the mantra of stay at home to 'save the NHS' was everywhere and across the world the same story was being sold when it was all lies. Two California doctors, Dan Erickson and Artin Massihi at Accelerated Urgent Care in Bakersfield, held a news conference in April, 2020, to say that intensive care units in California were 'empty, essentially', with hospitals shutting floors, not treating patients and laying off doctors. The California health system was working at minimum capacity 'getting rid of doctors because we just don't have the volume'. They said that people with conditions such as heart disease and cancer were not coming to hospital out of fear of 'Covid-19'. Their video was deleted by Susan Wojcicki's Cult-owned YouTube after reaching five million views. Florida governor Ron Desantis, who rejected the severe lockdowns of other states and is being targeted for doing so, said that in March, 2020, every US governor was given models claiming they would run out of hospital beds in days. That was never going to happen and the 'modellers' knew it. Deceit can be found at every level of the system. Urgent children's operations were cancelled including fracture repairs and biopsies to spot cancer. Eric Nicholls, a consultant paediatrician, said 'this is obviously concerning and we need to return to normal operating and to increase capacity as soon as possible'. Psychopaths

in power were rather less concerned *because* they are psychopaths. Deletion of urgent care and diagnosis has been happening all over the world and how many kids and others have died as a result of the actions of these cold and heartless lunatics dictating 'health' policy? The number must be stratospheric. Richard Sullivan, professor of cancer and global health at King's College London, said people feared 'Covid' more than cancer such was the campaign of fear. 'Years of lost life will be quite dramatic', Sullivan said, with 'a huge amount of avoidable mortality'. Sarah Woolnough, executive director for policy at Cancer Research UK, said there had been a 75 percent drop in urgent referrals to hospitals by family doctors of people with suspected cancer. Sullivan said that 'a lot of services have had to scale back – we've seen a dramatic decrease in the amount of elective cancer surgery'. Lockdown deaths worldwide has been absolutely fantastic with the *New York Post* reporting how data confirmed that 'lockdowns end more lives than they save':

> There was a sharp decline in visits to emergency rooms and an increase in fatal heart attacks because patients didn't receive prompt treatment. Many fewer people were screened for cancer. Social isolation contributed to excess deaths from dementia and Alzheimer's.
>
> Researchers predicted that the social and economic upheaval would lead to tens of thousands of "deaths of despair" from drug overdoses, alcoholism and suicide. As unemployment surged and mental-health and substance-abuse treatment programs were interrupted, the reported levels of anxiety, depression and suicidal thoughts increased dramatically, as did alcohol sales and fatal drug overdoses.

This has been happening while nurses and other staff had so much time on their hands in the 'war-zones' that Tic-Tok dancing videos began appearing across the Internet with medical staff dancing around in empty wards and corridors as people died at home from causes that would normally have been treated in hospital.

Mentions in dispatches

One brave and truth-committed whistleblower was Louise Hampton, a call handler with the UK NHS who made a viral Internet video saying she had done 'fuck all' during the 'pandemic'

which was 'a load of bollocks'. She said that 'Covid-19' was rebranded flu and of course she lost her job. This is what happens in the medical and endless other professions now when you tell the truth. Louise filmed inside 'war-zone' accident and emergency departments to show they were empty and I mean *empty* as in no one there. The mainstream media could have done the same and blown the gaff on the whole conspiracy. They haven't to their eternal shame. Not that most 'journalists' seem capable of manifesting shame as with the psychopaths they slavishly repeat without question. The relative few who were admitted with serious health problems were left to die alone with no loved ones allowed to see them because of 'Covid' rules and they included kids dying without the comfort of mum and dad at their bedside while the evil behind this couldn't give a damn. It was all good fun to them. A Scottish NHS staff nurse publicly quit in the spring of 2021 saying: 'I can no longer be part of the lies and the corruption by the government.' She said hospitals 'aren't full, the beds aren't full, beds have been shut, wards have been shut'. Hospitals were never busy throughout 'Covid'. The staff nurse said that Nicola Sturgeon, tragically the leader of the Scottish government, was on television saying save the hospitals and the NHS – 'but the beds are empty' and 'we've not seen flu, we always see flu every year'. She wrote to government and spoke with her union Unison (the unions are Cult-compromised and *useless*, but nothing changed. Many of her colleagues were scared of losing their jobs if they spoke out as they wanted to. She said nursing staff were being affected by wearing masks all day and 'my head is splitting every shift from wearing a mask'. The NHS is part of the fascist tyranny and must be dismantled so we can start again with human beings in charge. (Ironically, hospitals were reported to be busier again when official 'Covid' cases *fell* in spring/summer of 2021 and many other conditions required treatment at the same time as *the fake vaccine rollout*.)

I will cover the 'Covid vaccine' scam in detail later, but it is another indicator of the sickening disregard for human life that I am highlighting here. The DNA-manipulating concoctions do not fulfil

the definition of a 'vaccine', have never been used on humans before and were given only emergency approval because trials were not completed and they continued using the unknowing public. The result was what a NHS senior nurse with responsibility for 'vaccine' procedure said was 'genocide'. She said the 'vaccines' were not 'vaccines'. They had not been shown to be safe and claims about their effectiveness by drug companies were 'poetic licence'. She described what was happening as a 'horrid act of human annihilation'. The nurse said that management had instigated a policy of not providing a Patient Information Leaflet (PIL) before people were 'vaccinated' even though health care professionals are supposed to do this according to protocol. Patients should also be told that they are taking part in an ongoing clinical trial. Her challenges to what is happening had seen her excluded from meetings and ridiculed in others. She said she was told to 'watch my step … or I would find myself surplus to requirements'. The nurse, who spoke anonymously in fear of her career, said she asked her NHS manager why he/she was content with taking part in genocide against those having the 'vaccines'. The reply was that everyone had to play their part and to 'put up, shut up, and get it done'. Government was 'leaning heavily' on NHS management which was clearly leaning heavily on staff. This is how the global 'medical' hierarchy operates and it starts with the Cult and its World Health Organization.

She told the story of a doctor who had the Pfizer jab and when questioned had no idea what was in it. The doctor had never read the literature. We have to stop treating doctors as intellectual giants when so many are moral and medical pygmies. The doctor did not even know that the 'vaccines' were not fully approved or that their trials were ongoing. They were, however, asking their patients if they minded taking part in follow-ups for research purposes – yes, the *ongoing clinical trial*. The nurse said the doctor's ignorance was not rare and she had spoken to a hospital consultant who had the jab without any idea of the background or that the 'trials' had not been completed. Nurses and pharmacists had shown the same ignorance.

'My NHS colleagues have forsaken their duty of care, broken their code of conduct – Hippocratic Oath – and have been brainwashed just the same as the majority of the UK public through propaganda …' She said she had not been able to recruit a single NHS colleague, doctor, nurse or pharmacist to stand with her and speak out. Her union had refused to help. She said that if the genocide came to light she would not hesitate to give evidence at a Nuremberg-type trial against those in power who could have affected the outcomes but didn't.

And all for what?

To put the nonsense into perspective let's say the 'virus' does exist and let's go completely crazy and accept that the official manipulated figures for cases and deaths are accurate. *Even then* a study by Stanford University epidemiologist Dr John Ioannidis published on the World Health Organization website produced an average infection to fatality rate of … *0.23 percent!* Ioannidis said: 'If one could sample equally from all locations globally, the median infection fatality rate might even be substantially lower than the 0.23% observed in my analysis.' For healthy people under 70 it was … *0.05 percent*! This compares with the 3.4 percent claimed by the Cult-owned World Health Organization when the hoax was first played and maximum fear needed to be generated. An updated Stanford study in April, 2021, put the 'infection' to 'fatality' rate at just 0.15 percent. Another team of scientists led by Megan O'Driscoll and Henrik Salje studied data from 45 countries and published their findings on the Nature website. For children and young people the figure is so small it virtually does not register although authorities will be hyping dangers to the young when they introduce DNA-manipulating 'vaccines' for children. The O'Driscoll study produced an average infection-fatality figure of 0.003 for children from birth to four; 0.001 for 5 to 14; 0.003 for 15 to 19; and it was still only 0.456 up to 64. To claim that children must be 'vaccinated' to protect them from 'Covid' is an obvious lie and so there must be another reason and there is. What's more the average age of a 'Covid' death is akin

to the average age that people die in general. The average age of death in England is about 80 for men and 83 for women. The average age of death from alleged 'Covid' is between 82 and 83. California doctors, Dan Erickson and Artin Massihi, said at their April media conference that projection models of millions of deaths had been 'woefully inaccurate'. They produced detailed figures showing that Californians had a 0.03 chance of dying from 'Covid' based on the number of people who tested positive (with a test not testing for the 'virus'). Erickson said there was a 0.1 percent chance of dying from 'Covid' in the *state* of New York, not just the city, and a 0.05 percent chance in Spain, a centre of 'Covid-19' hysteria at one stage. The Stanford studies supported the doctors' data with fatality rate estimates of 0.23 and 0.15 percent. How close are these figures to my estimate of *zero*? Death-rate figures claimed by the World Health Organization at the start of the hoax were some 15 times higher. The California doctors said there was no justification for lockdowns and the economic devastation they caused. Everything they had ever learned about quarantine was that you quarantine the *sick* and not the healthy. They had never seen this before and it made no medical sense.

Why in the in the light of all this would governments and medical systems the world over say that billions must go under house arrest; lose their livelihood; in many cases lose their mind, their health and their life; force people to wear masks dangerous to health and psychology; make human interaction and even family interaction a criminal offence; ban travel; close restaurants, bars, watching live sport, concerts, theatre, and any activity involving human togetherness and discourse; and closing schools to isolate children from their friends and cause many to commit suicide in acts of hopelessness and despair? The California doctors said lockdown consequences included increased child abuse, partner abuse, alcoholism, depression, and other impacts they were seeing every day. Who would do that to the entire human race if not mentally-ill psychopaths of almost unimaginable extremes like Bill Gates? We must face the reality of what we are dealing with and come out of

denial. Fascism and tyranny are made possible only by the target population submitting and acquiescing to fascism and tyranny. The whole of human history shows that to be true. Most people naively and unquestioning believed what they were told about a 'deadly virus' and meekly and weakly submitted to house arrest. Those who didn't believe it – at least in total – still submitted in fear of the consequences of not doing so. For the rest who wouldn't submit draconian fines have been imposed, brutal policing by psychopaths *for* psychopaths, and condemnation from the meek and weak who condemn the Pushbackers on behalf of the very force that has them, too, in its gunsights. 'Pathetic' does not even begin to suffice. Britain's brainless 'Health' Secretary Matt Hancock warned anyone lying to border officials about returning from a list of 'hotspot' countries could face a jail sentence of up to ten years which is more than for racially-aggravated assault, incest and attempting to have sex with a child under 13. Hancock is a lunatic, but he has the state apparatus behind him in a Cult-led chain reaction and the same with UK 'Vaccine Minister' Nadhim Zahawi, a prominent member of the mega-Cult secret society, Le Cercle, which featured in my earlier books. The Cult enforces its will on governments and medical systems; government and medical systems enforce their will on business and police; business enforces its will on staff who enforce it on customers; police enforce the will of the Cult on the population and play their essential part in creating a world of fascist control that their own children and grandchildren will have to live in their entire lives. It is a hierarchical pyramid of imposition and acquiescence and, yes indeedy, of clinical insanity.

Does anyone bright enough to read this book have to ask what the answer is? I think not, but I will reveal it anyway in the fewest of syllables: Tell the psychos and their moronic lackeys to fuck off and let's get on with our lives. We are many – They are few.

CHAPTER SEVEN

War on your mind

One believes things because one has been conditioned to believe them
Aldous Huxley, *Brave New World*

I have described the 'Covid' hoax as a 'Psyop' and that is true in every sense and on every level in accordance with the definition of that term which is psychological warfare. Break down the 'Covid pandemic' to the foundation themes and it is psychological warfare on the human individual and collective mind.

The same can be said for the entire human belief system involving every subject you can imagine. Huxley was right in his contention that people believe what they are conditioned to believe and this comes from the repetition throughout their lives of the same falsehoods. They spew from government, corporations, media and endless streams of 'experts' telling you what the Cult wants you to believe and often believing it themselves (although *far* from always). 'Experts' are rewarded with 'prestigious' jobs and titles and as agents of perceptual programming with regular access to the media. The Cult has to control the narrative – control *information* – or they lose control of the vital, crucial, without-which-they-cannot-prevail public perception of reality. The foundation of that control today is the Internet made possible by the Defense Advanced Research Projects Agency (DARPA), the incredibly sinister technological arm of the Pentagon. The Internet is the result of military technology.

DARPA openly brags about establishing the Internet which has been a long-term project to lasso the minds of the global population. I have said for decades the plan is to control information to such an extreme that eventually no one would see or hear anything that the Cult does not approve. We are closing in on that end with ferocious censorship since the 'Covid' hoax began and in my case it started back in the 1990s in terms of books and speaking venues. I had to create my own publishing company in 1995 precisely because no one else would publish my books even then. I think they're all still running.

Cult Internet

To secure total control of information they needed the Internet in which pre-programmed algorithms can seek out 'unclean' content for deletion and even stop it being posted in the first place. The Cult had to dismantle print and non-Internet broadcast media to ensure the transfer of information to the appropriate-named 'Web' – a critical expression of the *Cult* web. We've seen the ever-quickening demise of traditional media and control of what is left by a tiny number of corporations operating worldwide. Independent journalism in the mainstream is already dead and never was that more obvious than since the turn of 2020. The Cult wants all information communicated via the Internet to globally censor and allow the plug to be pulled any time. Lockdowns and forced isolation has meant that communication between people has been through electronic means and no longer through face-to-face discourse and discussion. Cult psychopaths have targeted the bars, restaurants, sport, venues and meeting places in general for this reason. None of this is by chance and it's to stop people gathering in any kind of privacy or number while being able to track and monitor all Internet communications and block them as necessary. Even private messages between individuals have been censored by these fascists that control Cult fronts like Facebook, Twitter, Google and YouTube which are all officially run by Sabbatian place-people and from the background by higher-level Sabbatian place people.

Facebook, Google, Amazon and their like were seed-funded and supported into existence with money-no-object infusions of funds either directly or indirectly from DARPA and CIA technology arm In-Q-Tel. The Cult plays the long game and prepares very carefully for big plays like 'Covid'. Amazon is another front in the psychological war and pretty much controls the global market in book sales and increasingly publishing. Amazon's limitless funds have deleted fantastic numbers of independent publishers to seize global domination on the way to deciding which books can be sold and circulated and which cannot. Moves in that direction are already happening. Amazon's leading light Jeff Bezos is the grandson of Lawrence Preston Gise who worked with DARPA predecessor ARPA. Amazon has big connections to the CIA and the Pentagon. The plan I have long described went like this:

1. Employ military technology to establish the Internet.

2. Sell the Internet as a place where people can freely communicate without censorship and allow that to happen until the Net becomes the central and irreversible pillar of human society. If the Internet had been highly censored from the start many would have rejected it.

3. Fund and manipulate major corporations into being to control the circulation of information on your Internet using cover stories about geeks in garages to explain how they came about. Give them unlimited funds to expand rapidly with no need to make a profit for years while non-Cult companies who need to balance the books cannot compete. You know that in these circumstances your Googles, YouTubes, Facebooks and Amazons are going to secure near monopolies by either crushing or buying up the opposition.

4. Allow freedom of expression on both the Internet and communication platforms to draw people in until the Internet is the central and irreversible pillar of human society and your communication corporations have reached a stage of near monopoly domination.

5. Then unleash your always-planned frenzy of censorship on the basis of 'where else are you going to go?' and continue to expand that until nothing remains that the Cult does not want its human targets to see.

The process was timed to hit the 'Covid' hoax to ensure the best chance possible of controlling the narrative which they knew they had to do at all costs. They were, after all, about to unleash a 'deadly virus' that didn't really exist. If you do that in an environment of free-flowing information and opinion you would be dead in the

water before you could say Gates is a psychopath. The network was in place through which the Cult-created-and-owned World Health Organization could dictate the 'Covid' narrative and response policy slavishly supported by Cult-owned Internet communication giants and mainstream media while those telling a different story were censored. Google, YouTube, Facebook and Twitter openly announced that they would do this. What else would we expect from Cult-owned operations like Facebook which former executives have confirmed set out to make the platform more addictive than cigarettes and coldly manipulates emotions of its users to sow division between people and groups and scramble the minds of the young? If Zuckerberg lives out the rest of his life without going to jail for crimes against humanity, and most emphatically against the young, it will be a travesty of justice. Still, no matter, cause and effect will catch up with him eventually and the same with Sergey Brin and Larry Page at Google with its CEO Sundar Pichai who fix the Google search results to promote Cult narratives and hide the opposition. Put the same key words into Google and other search engines like DuckDuckGo and you will see how different results can be. Wikipedia is another intensely biased 'encyclopaedia' which skews its content to the Cult agenda. YouTube links to Wikipedia's version of 'Covid' and 'climate change' on video pages in which experts in their field offer a different opinion (even that is increasingly rare with Wojcicki censorship). Into this 'Covid' silence-them network must be added government media censors, sorry 'regulators', such as Ofcom in the UK which imposed tyrannical restrictions on British broadcasters that had the effect of banning me from ever appearing. Just to debate with me about my evidence and views on 'Covid' would mean breaking the fascistic impositions of Ofcom and its CEO career government bureaucrat Melanie Dawes. Gutless British broadcasters tremble at the very thought of fascist Ofcom.

Psychos behind 'Covid'

The reason for the 'Covid' catastrophe in all its facets and forms can be seen by whom and what is driving the policies worldwide in such a coordinated way. Decisions are not being made to protect health, but to target psychology. The dominant group guiding and 'advising' government policy are not medical professionals. They are psychologists and behavioural scientists. Every major country has its own version of this phenomenon and I'll use the British example to show how it works. In many ways the British version has been affecting the wider world in the form of the huge behaviour manipulation network in the UK which operates in other countries. The network involves private companies, government, intelligence and military. The Cabinet Office is at the centre of the government 'Covid' Psyop and part-owns, with 'innovation charity' Nesta, the Behavioural Insights Team (BIT) which claims to be independent of government but patently isn't. The BIT was established in 2010 and its job is to manipulate the psyche of the population to acquiesce to government demands and so much more. It is also known as the 'Nudge Unit', a name inspired by the 2009 book by two ultra-Zionists, Cass Sunstein and Richard Thaler, called *Nudge: Improving Decisions About Health, Wealth, and Happiness*. The book, as with the Behavioural Insights Team, seeks to 'nudge' behaviour (manipulate it) to make the public follow patterns of action and perception that suit those in authority (the Cult). Sunstein is so skilled at this that he advises the World Health Organization and the UK Behavioural Insights Team and was Administrator of the White House Office of Information and Regulatory Affairs in the Obama administration. Biden appointed him to the Department of Homeland Security – another ultra-Zionist in the fold to oversee new immigration laws which is another policy the Cult wants to control. Sunstein is desperate to silence anyone exposing conspiracies and co-authored a 2008 report on the subject in which suggestions were offered to ban 'conspiracy theorizing' or impose 'some kind of tax, financial or otherwise, on those who disseminate such theories'. I guess a psychiatrist's chair is out of the question?

Sunstein's mate Richard Thaler, an 'academic affiliate' of the UK Behavioural Insights Team, is a proponent of 'behavioural economics' which is defined as the study of 'the effects of psychological, cognitive, emotional, cultural and social factors on the decisions of individuals and institutions'. Study the effects so they can be manipulated to be what you want them to be. Other leading names in the development of behavioural economics are ultra-Zionists Daniel Kahneman and Robert J. Shiller and they, with Thaler, won the Nobel Memorial Prize in Economic Sciences for their work in this field. The Behavioural Insights Team is operating at the heart of the UK government and has expanded globally through partnerships with several universities including Harvard, Oxford, Cambridge, University College London (UCL) and Pennsylvania. They claim to have 'trained' (reframed) 20,000 civil servants and run more than 750 projects involving 400 randomised controlled trials in dozens of countries' as another version of mind reframers Common Purpose. BIT works from its office in New York with cities and their agencies, as well as other partners, across the United States and Canada – this is a company part-owned by the British government Cabinet Office. An executive order by President Cult-servant Obama established a US Social and Behavioral Sciences Team in 2015. They all have the same reason for being and that's to brainwash the population directly and by brainwashing those in positions of authority.

'Covid' mind game

Another prime aspect of the UK mind-control network is the 'independent' [joke] Scientific Pandemic Insights Group on Behaviours (SPI-B) which 'provides behavioural science advice aimed at anticipating and helping people adhere to interventions that are recommended by medical or epidemiological experts'. That means manipulating public perception and behaviour to do whatever government tells them to do. It's disgusting and if they really want the public to be 'safe' this lot should all be under lock and key. According to the government website SPI-B consists of

'behavioural scientists, health and social psychologists, anthropologists and historians' and advises the Whitty-Vallance-led Scientific Advisory Group for Emergencies (SAGE) which in turn advises the government on 'the science' (it doesn't) and 'Covid' policy. When politicians say they are being guided by 'the science' this is the rabble in each country they are talking about and that 'science' is dominated by behaviour manipulators to enforce government fascism through public compliance. The Behaviour Insight Team is headed by psychologist David Solomon Halpern, a visiting professor at King's College London, and connects with a national and global web of other civilian and military organisations as the Cult moves towards its goal of fusing them into one fascistic whole in every country through its 'Fusion Doctrine'. The behaviour manipulation network involves, but is not confined to, the Foreign Office; National Security Council; government communications headquarters (GCHQ); MI5; MI6; the Cabinet Office-based Media Monitoring Unit; and the Rapid Response Unit which 'monitors digital trends to spot emerging issues; including misinformation and disinformation; and identifies the best way to respond'.

There is also the 77th Brigade of the UK military which operates like the notorious Israeli military's Unit 8200 in manipulating information and discussion on the Internet by posing as members of the public to promote the narrative and discredit those who challenge it. Here we have the military seeking to manipulate *domestic* public opinion while the Nazis in government are fine with that. Conservative Member of Parliament Tobias Ellwood, an advocate of lockdown and control through 'vaccine passports', is a Lieutenant Colonel reservist in the 77th Brigade which connects with the military operation jHub, the 'innovation centre' for the Ministry of Defence and Strategic Command. jHub has also been involved with the civilian National Health Service (NHS) in 'symptom tracing' the population. The NHS is a key part of this mind control network and produced a document in December, 2020, explaining to staff how to use psychological manipulation with different groups and ages to get them to have the DNA-manipulating 'Covid vaccine'

that's designed to cumulatively rewrite human genetics. The document, called 'Optimising Vaccination Roll Out – Do's and Dont's for all messaging, documents and "communications" in the widest sense', was published by NHS England and the NHS Improvement *Behaviour Change Unit* in partnership with Public Health England and Warwick Business School. I hear the mantra about 'save the NHS' and 'protect the NHS' when we need to scrap the NHS and start again. The current version is far too corrupt, far too anti-human and totally compromised by Cult operatives and their assets. UK government broadcast media censor Ofcom will connect into this web – as will the BBC with its tremendous Ofcom influence – to control what the public see and hear and dictate mass perception. Nuremberg trials must include personnel from all these organisations.

The fear factor

The 'Covid' hoax has led to the creation of the UK Cabinet Office-connected Joint Biosecurity Centre (JBC) which is officially described as providing 'expert advice on pandemics' using its independent [all Cult operations are 'independent'] analytical function to provide real-time analysis about infection outbreaks to identify and respond to outbreaks of Covid-19'. Another role is to advise the government on a response to spikes in infections – 'for example by closing schools or workplaces in local areas where infection levels have risen'. Put another way, promoting the Cult agenda. The Joint Biosecurity Centre is modelled on the Joint Terrorism Analysis Centre which analyses intelligence to set 'terrorism threat levels' and here again you see the fusion of civilian and military operations and intelligence that has led to military intelligence producing documents about 'vaccine hesitancy' and how it can be combated. Domestic civilian matters and opinions should not be the business of the military. The Joint Biosecurity Centre is headed by Tom Hurd, director general of the Office for Security and Counter-Terrorism from the establishment-to-its-fingertips Hurd family. His father is former Foreign Secretary Douglas Hurd. How coincidental that Tom

Hurd went to the elite Eton College and Oxford University with Boris Johnson. Imperial College with its ridiculous computer modeller Neil Ferguson will connect with this gigantic web that will itself interconnect with similar set-ups in other major and not so major countries. Compared with this Cult network the politicians, be they Boris Johnson, Donald Trump or Joe Biden, are bit-part players 'following the science'. The network of psychologists was on the 'Covid' case from the start with the aim of generating maximum fear of the 'virus' to ensure compliance by the population. A government behavioural science group known as SPI-B produced a paper in March, 2020, for discussion by the main government science advisory group known as SAGE. It was headed 'Options for increasing adherence to social distancing measures' and it said the following in a section headed 'Persuasion':

- A substantial number of people still do not feel sufficiently personally threatened; it could be that they are reassured by the low death rate in their demographic group, although levels of concern may be rising. Having a good understanding of the risk has been found to be positively associated with adoption of COVID-19 social distancing measures in Hong Kong.

- The perceived level of personal threat needs to be increased among those who are complacent, using hard-hitting evaluation of options for increasing social distancing emotional messaging. To be effective this must also empower people by making clear the actions they can take to reduce the threat.

- Responsibility to others: There seems to be insufficient understanding of, or feelings of responsibility about, people's role in transmitting the infection to others … Messaging about actions need to be framed positively in terms of protecting oneself and the community, and increase confidence that they will be effective.

- Some people will be more persuaded by appeals to play by the rules, some by duty to the community, and some to personal risk.

> All these different approaches are needed. The messaging also needs to take account of the realities of different people's lives. Messaging needs to take account of the different motivational levers and circumstances of different people.

All this could be achieved the SPI-B psychologists said by *using the media to increase the sense of personal threat* which translates as terrify the shit out of the population, including children, so they all do what we want. That's not happened has it? Those excuses for 'journalists' who wouldn't know journalism if it bit them on the arse (the great majority) have played their crucial part in serving this Cult-government Psyop to enslave their own kids and grandkids. How they live with themselves I have no idea. The psychological war has been underpinned by constant government 'Covid' propaganda in almost every television and radio ad break, plus the Internet and print media, which has pounded out the fear with taxpayers footing the bill for their own programming. The result has been people terrified of a 'virus' that doesn't exist or one with a tiny fatality rate even if you believe it does. People walk down the street and around the shops wearing face-nappies damaging their health and psychology while others report those who refuse to be that naïve to the police who turn up in their own face-nappies. I had a cameraman come to my flat and he was so frightened of 'Covid' he came in wearing a mask and refused to shake my hand in case he caught something. He had – naïveitis – and the thought that he worked in the mainstream media was both depressing and made his behaviour perfectly explainable. The fear which has gripped the minds of so many and frozen them into compliance has been carefully cultivated by these psychologists who are really psychopaths. If lives get destroyed and a lot of young people commit suicide it shows our plan is working. SPI-B then turned to compulsion on the public to comply. 'With adequate preparation, rapid change can be achieved', it said. Some countries had introduced mandatory self-isolation on a wide scale without evidence of major public unrest and a large majority of the UK's population appeared to be supportive of more coercive measures with 64 percent of adults saying they would

support putting London under a lockdown (watch the 'polls' which are designed to make people believe that public opinion is in favour or against whatever the subject in hand).

For 'aggressive protective measures' to be effective, the SPI-B paper said, special attention should be devoted to those population groups that are more at risk. Translated from the Orwellian this means making the rest of population feel guilty for not protecting the 'vulnerable' such as old people which the Cult and its agencies were about to kill on an industrial scale with lockdown, lack of treatment and the Gates 'vaccine'. Psychopath psychologists sold their guilt-trip so comprehensively that Los Angeles County Supervisor Hilda Solis reported that children were apologising (from a distance) to their parents and grandparents for bringing 'Covid' into their homes and getting them sick. '… These apologies are just some of the last words that loved ones will ever hear as they die alone,' she said. Gut-wrenchingly Solis then used this childhood tragedy to tell children to stay at home and 'keep your loved ones alive'. Imagine heaping such potentially life-long guilt on a kid when it has absolutely nothing to do with them. These people are deeply disturbed and the psychologists behind this even more so.

Uncivil war – divide and rule

Professional mind-controllers at SPI-B wanted the media to increase a sense of responsibility to others (do as you're told) and promote 'positive messaging' for those actions while in contrast to invoke 'social disapproval' by the unquestioning, obedient, community of anyone with a mind of their own. Again the compliant Goebbels-like media obliged. This is an old, old, trick employed by tyrannies the world over throughout human history. You get the target population to keep the target population in line – *your* line. SPI-B said this could 'play an important role in preventing anti-social behaviour or discouraging failure to enact pro-social behaviour'. For 'anti-social' in the Orwellian parlance of SPI-B see any behaviour that government doesn't approve. SPI-B recommendations said that 'social disapproval' should be accompanied by clear messaging and

promotion of strong collective identity – hence the government and celebrity mantra of 'we're all in this together'. Sure we are. The mind doctors have such contempt for their targets that they think some clueless comedian, actor or singer telling them to do what the government wants will be enough to win them over. We have had UK comedian Lenny Henry, actor Michael Caine and singer Elton John wheeled out to serve the propagandists by urging people to have the DNA-manipulating 'Covid' non-'vaccine'. The role of Henry and fellow black celebrities in seeking to coax a 'vaccine' reluctant black community into doing the government's will was especially stomach-turning. An emotion-manipulating script and carefully edited video featuring these black 'celebs' was such an insult to the intelligence of black people and where's the self-respect of those involved selling their souls to a fascist government agenda? Henry said he heard black people's 'legitimate worries and concerns', but people must 'trust the facts' when they were doing exactly that by not having the 'vaccine'. They had to include the obligatory reference to Black Lives Matter with the line … 'Don't let coronavirus cost even more black lives – because we matter'. My god, it was pathetic. 'I know the vaccine is safe and what it does.' How? 'I'm a comedian and it says so in my script.'

SPI-B said social disapproval needed to be carefully managed to avoid victimisation, scapegoating and misdirected criticism, but they knew that their 'recommendations' would lead to exactly that and the media were specifically used to stir-up the divide-and-conquer hostility. Those who conform like good little baa, baas, are praised while those who have seen through the tidal wave of lies are 'Covidiots'. The awake have been abused by the fast asleep for not conforming to fascism and impositions that the awake know are designed to endanger their health, dehumanise them, and tear asunder the very fabric of human society. We have had the curtain-twitchers and morons reporting neighbours and others to the face-nappied police for breaking 'Covid rules' with fascist police delighting in posting links and phone numbers where this could be done. The Cult cannot impose its will without a compliant police

and military or a compliant population willing to play their part in enslaving themselves and their kids. The words of a pastor in Nazi Germany are so appropriate today:

First they came for the socialists and I did not speak out because I was not a socialist.

Then they came for the trade unionists and I did not speak out because I was not a trade unionist.

Then they came for the Jews and I did not speak out because I was not a Jew.

Then they came for me and there was no one left to speak for me.

Those who don't learn from history are destined to repeat it and so many are.

'Covid' rules: Rewiring the mind

With the background laid out to this gigantic national and global web of psychological manipulation we can put 'Covid' rules into a clear and sinister perspective. Forget the claims about protecting health. 'Covid' rules are about dismantling the human mind, breaking the human spirit, destroying self-respect, and then putting Humpty Dumpty together again as a servile, submissive slave. Social isolation through lockdown and distancing have devastating effects on the human psyche as the psychological psychopaths well know and that's the real reason for them. Humans need contact with each other, discourse, closeness and touch, or they eventually, and literarily, go crazy. Masks, which I will address at some length, fundamentally add to the effects of isolation and the Cult agenda to dehumanise and de-individualise the population. To do this while knowing – in fact *seeking* – this outcome is the very epitome of evil and psychologists involved in this *are* the epitome of evil. They must like all the rest of the Cult demons and their assets stand trial for crimes against humanity on a scale that defies the imagination. Psychopaths in uniform use isolation to break enemy troops and agents and make them subservient and submissive to tell what they know. The technique is rightly considered a form of torture and

torture is most certainly what has been imposed on the human population.

Clinically-insane American psychologist Harry Harlow became famous for his isolation experiments in the 1950s in which he separated baby monkeys from their mothers and imprisoned them for months on end in a metal container or 'pit of despair'. They soon began to show mental distress and depression as any idiot could have predicted. Harlow put other monkeys in steel chambers for three, six or twelve months while denying them any contact with animals or humans. He said that the effects of total social isolation for six months were 'so devastating and debilitating that we had assumed initially that twelve months of isolation would not produce any additional decrement'; but twelve months of isolation 'almost obliterated the animals socially'. This is what the Cult and its psychopaths are doing to you and your children. Even monkeys in partial isolation in which they were not allowed to form relationships with other monkeys became 'aggressive and hostile, not only to others, but also towards their own bodies'. We have seen this in the young as a consequence of lockdown. UK government psychopaths launched a public relations campaign telling people not to hug each other even after they received the 'Covid-19 vaccine' which we were told with more lies would allow a return to 'normal life'. A government source told *The Telegraph*: 'It will be along the lines that it is great that you have been vaccinated, but if you are going to visit your family and hug your grandchildren there is a chance you are going to infect people you love.' The source was apparently speaking from a secure psychiatric facility. Janet Lord, director of Birmingham University's Institute of Inflammation and Ageing, said that parents and grandparents should avoid hugging their children. Well, how can I put it, Ms Lord? Fuck off. Yep, that'll do.

Destroying the kids – where are the parents?

Observe what has happened to people enslaved and isolated by lockdown as suicide and self-harm has soared worldwide,

particularly among the young denied the freedom to associate with their friends. A study of 49,000 people in English-speaking countries concluded that almost half of young adults are at clinical risk of mental health disorders. A national survey in America of 1,000 currently enrolled high school and college students found that 5 percent reported attempting suicide during the pandemic. Data from the US CDC's National Syndromic Surveillance Program from January 1st to October 17th, 2020, revealed a *31 percent* increase in mental health issues among adolescents aged 12 to 17 compared with 2019. The CDC reported that America in general suffered the biggest drop in life expectancy since World War Two as it fell by a year in the first half of 2020 as a result of 'deaths of despair' – overdoses and suicides. Deaths of despair have leapt by more than 20 percent during lockdown and include the highest number of fatal overdoses ever recorded in a single year – 81,000. Internet addiction is another consequence of being isolated at home which lowers interest in physical activities as kids fall into inertia and what's the point? Children and young people are losing hope and giving up on life, sometimes literally. A 14-year-old boy killed himself in Maryland because he had 'given up' when his school district didn't reopen; an 11-year-old boy shot himself during a zoom class; a teenager in Maine succumbed to the isolation of the 'pandemic' when he ended his life after experiencing a disrupted senior year at school. Children as young as nine have taken their life and all these stories can be repeated around the world. Careers are being destroyed before they start and that includes those in sport in which promising youngsters have not been able to take part. The plan of the psycho-psychologists is working all right. Researchers at Cambridge University found that lockdowns cause significant harm to children's mental health. Their study was published in the *Archives of Disease in Childhood*, and followed 168 children aged between 7 and 11. The researchers concluded:

> During the UK lockdown, children's depression symptoms have increased substantially, relative to before lockdown. The scale of this effect has direct relevance for the continuation of different elements of lockdown policy, such as complete or partial school closures …

… Specifically, we observed a statistically significant increase in ratings of depression, with a medium-to-large effect size. Our findings emphasise the need to incorporate the potential impact of lockdown on child mental health in planning the ongoing response to the global pandemic and the recovery from it.

Not a chance when the Cult's psycho-psychologists were getting exactly what they wanted. The UK's Royal College of Paediatrics and Child Health has urged parents to look for signs of eating disorders in children and young people after a three to four fold increase. Specialists say the 'pandemic' is a major reason behind the rise. You don't say. The College said isolation from friends during school closures, exam cancellations, loss of extra-curricular activities like sport, and an increased use of social media were all contributory factors along with fears about the virus (psycho-psychologists again), family finances, and students being forced to quarantine. Doctors said young people were becoming severely ill by the time they were seen with 'Covid' regulations reducing face-to-face consultations. Nor is it only the young that have been devastated by the psychopaths. Like all bullies and cowards the Cult is targeting the young, elderly, weak and infirm. A typical story was told by a British lady called Lynn Parker who was not allowed to visit her husband in 2020 for the last ten and half months of his life 'when he needed me most' between March 20th and when he died on December 19th. This vacates the criminal and enters the territory of evil. The emotional impact on the immune system alone is immense as are the number of people of all ages worldwide who have died as a result of Cult-demanded, Gates-demanded, lockdowns.

Isolation is torture

The experience of imposing solitary confinement on millions of prisoners around the world has shown how a large percentage become 'actively psychotic and/or acutely suicidal'. Social isolation has been found to trigger 'a specific psychiatric syndrome, characterized by hallucinations; panic attacks; overt paranoia; diminished impulse control; hypersensitivity to external stimuli; and difficulties with thinking, concentration and memory'. Juan Mendez,

a United Nations rapporteur (investigator), said that isolation is a form of torture. Research has shown that even after isolation prisoners find it far more difficult to make social connections and I remember chatting to a shop assistant after one lockdown who told me that when her young son met another child again he had no idea how to act or what to do. Hannah Flanagan, Director of Emergency Services at Journey Mental Health Center in Dane County, Wisconsin, said: 'The specificity about Covid social distancing and isolation that we've come across as contributing factors to the suicides are really new to us this year.' But they are not new to those that devised them. They are getting the effect they want as the population is psychologically dismantled to be rebuilt in a totally different way. Children and the young are particularly targeted. They will be the adults when the full-on fascist AI-controlled technocracy is planned to be imposed and they are being prepared to meekly submit. At the same time older people who still have a memory of what life was like before – and how fascist the new normal really is – are being deleted. You are going to see efforts to turn the young against the old to support this geriatric genocide. Hannah Flanagan said the big increase in suicide in her county proved that social isolation is not only harmful, but deadly. Studies have shown that isolation from others is one of the main risk factors in suicide and even more so with women. Warnings that lockdown could create a 'perfect storm' for suicide were ignored. After all this was one of the *reasons* for lockdown. Suicide, however, is only the most extreme of isolation consequences. There are many others. Dr Dhruv Khullar, assistant professor of healthcare policy at Weill Cornell Medical College, said in a *New York Times* article in 2016 long before the fake 'pandemic':

> A wave of new research suggests social separation is bad for us. Individuals with less social connection have disrupted sleep patterns, altered immune systems, more inflammation and higher levels of stress hormones. One recent study found that isolation increases the risk of heart disease by 29 percent and stroke by 32 percent. Another analysis that pooled data from 70 studies and 3.4 million people found that socially isolated individuals had a 30 percent higher risk of dying in the next seven years, and that this effect was largest in middle age.

Loneliness can accelerate cognitive decline in older adults, and isolated individuals are twice as likely to die prematurely as those with more robust social interactions. These effects start early: Socially isolated children have significantly poorer health 20 years later, even after controlling for other factors. All told, loneliness is as important a risk factor for early death as obesity and smoking.

There you have proof from that one article alone four years before 2020 that those who have enforced lockdown, social distancing and isolation knew what the effect would be and that is even more so with professional psychologists that have been driving the policy across the globe. We can go back even further to the years 2000 and 2003 and the start of a major study on the effects of isolation on health by Dr Janine Gronewold and Professor Dirk M. Hermann at the University Hospital in Essen, Germany, who analysed data on 4,316 people with an average age of 59 who were recruited for the long-term research project. They found that socially isolated people are more than 40 percent more likely to have a heart attack, stroke, or other major cardiovascular event and nearly 50 percent more likely to die from any cause. Given the financial Armageddon unleashed by lockdown we should note that the study found a relationship between increased cardiovascular risk and lack of financial support. After excluding other factors social isolation was still connected to a 44 percent increased risk of cardiovascular problems and a 47 percent increased risk of death by any cause. Lack of financial support was associated with a 30 percent increase in the risk of cardiovascular health events. Dr Gronewold said it had been known for some time that feeling lonely or lacking contact with close friends and family can have an impact on physical health and the study had shown that having strong social relationships is of high importance for heart health. Gronewold said they didn't understand yet why people who are socially isolated have such poor health outcomes, but this was obviously a worrying finding, particularly during these times of prolonged social distancing. Well, it can be explained on many levels. You only have to identify the point in the body where people feel loneliness and missing people they are parted from – it's in the centre of the chest where they feel the ache of loneliness and the ache of missing people. 'My heart aches for

you' … 'My heart aches for some company.' I will explain this more in the chapter Escaping Wetiko, but when you realise that the body is the mind – they are expressions of each other – the reason why state of the mind dictates state of the body becomes clear.

American psychologist Ranjit Powar was highlighting the effects of lockdown isolation as early as April, 2020. She said humans have evolved to be social creatures and are wired to live in interactive groups. Being isolated from family, friends and colleagues could be unbalancing and traumatic for most people and could result in short or even long-term psychological and physical health problems. An increase in levels of anxiety, aggression, depression, forgetfulness and hallucinations were possible psychological effects of isolation. 'Mental conditions may be precipitated for those with underlying pre-existing susceptibilities and show up in many others without any pre-condition.' Powar said personal relationships helped us cope with stress and if we lost this outlet for letting off steam the result can be a big emotional void which, for an average person, was difficult to deal with. 'Just a few days of isolation can cause increased levels of anxiety and depression' – so what the hell has been the effect on the global population of *18 months* of this at the time of writing? Powar said: 'Add to it the looming threat of a dreadful disease being repeatedly hammered in through the media and you have a recipe for many shades of mental and physical distress.' For those with a house and a garden it is easy to forget that billions have had to endure lockdown isolation in tiny overcrowded flats and apartments with nowhere to go outside. The psychological and physical consequences of this are unimaginable and with lunatic and abusive partners and parents the consequences have led to tremendous increases in domestic and child abuse and alcoholism as people seek to shut out the horror. Ranjit Powar said:

> Staying in a confined space with family is not all a rosy picture for everyone. It can be extremely oppressive and claustrophobic for large low-income families huddled together in small single-room houses. Children here are not lucky enough to have many board/electronic games or books to keep them occupied.

Add to it the deep insecurity of running out of funds for food and basic necessities. On the other hand, there are people with dysfunctional family dynamics, such as domineering, abusive or alcoholic partners, siblings or parents which makes staying home a period of trial. Incidence of suicide and physical abuse against women has shown a worldwide increase. Heightened anxiety and depression also affect a person's immune system, making them more susceptible to illness.

To think that Powar's article was published on April 11th, 2020.

Six-feet fantasy

Social (unsocial) distancing demanded that people stay six feet or two metres apart. UK government advisor Robert Dingwall from the New and Emerging Respiratory Virus Threats Advisory Group said in a radio interview that the two-metre rule was 'conjured up out of nowhere' and was not based on science. No, it was not based on *medical* science, but it didn't come out of nowhere. The distance related to *psychological* science. Six feet/two metres was adopted in many countries and we were told by people like the criminal Anthony Fauci and his ilk that it was founded on science. Many schools could not reopen because they did not have the space for six-feet distancing. Then in March, 2021, after a year of six-feet 'science', a study published in the *Journal of Infectious Diseases* involving more than 500,000 students and almost 100,000 staff over 16 weeks revealed no significant difference in 'Covid' cases between six feet and three feet and Fauci changed his tune. Now three feet was okay. There is no difference between six feet and three *inches* when there is no 'virus' and they got away with six feet for psychological reasons for as long as they could. I hear journalists and others talk about 'unintended consequences' of lockdown. They are not *unintended* at all; they have been coldly-calculated for a specific outcome of human control and that's why super-psychopaths like Gates have called for them so vehemently. Super-psychopath psychologists have demanded them and psychopathic or clueless, spineless, politicians have gone along with them by 'following the science'. But it's not science at all. 'Science' is not what is; it's only what people can be manipulated to believe it is. The whole 'Covid' catastrophe is

founded on mind control. Three word or three statement mantras issued by the UK government are a well-known mind control technique and so we've had 'Stay home/protect the NHS/save lives', 'Stay alert/control the virus/save lives' and 'hands/face/space'. One of the most vocal proponents of extreme 'Covid' rules in the UK has been Professor Susan Michie, a member of the British Communist Party, who is not a medical professional. Michie is the director of the Centre for Behaviour Change at University College London. She is a *behavioural psychologist* and another filthy rich 'Marxist' who praised China's draconian lockdown. She was known by fellow students at Oxford University as 'Stalin's nanny' for her extreme Marxism. Michie is an influential member of the UK government's Scientific Advisory Group for Emergencies (SAGE) and behavioural manipulation groups which have dominated 'Covid' policy. She is a consultant adviser to the World Health Organization on 'Covid-19' and behaviour. Why the hell are lockdowns anything to do with her when they are claimed to be about health? Why does a behavioural psychologist from a group charged with changing the behaviour of the public want lockdown, human isolation and mandatory masks? Does that question really need an answer? Michie *absolutely* has to explain herself before a Nuremberg court when humanity takes back its world again and even more so when you see the consequences of masks that she demands are compulsory. This is a Michie classic:

> The benefits of getting primary school children to wear masks is that regardless of what little degree of transmission is occurring in those age groups it could help normalise the practice. Young children wearing masks may be more likely to get their families to accept masks.

Those words alone should carry a prison sentence when you ponder on the callous disregard for children involved and what a statement it makes about the mind and motivations of Susan Michie. What a lovely lady and what she said there encapsulates the mentality of the psychopaths behind the 'Covid' horror. Let us compare what Michie said with a countrywide study in Germany published at researchsquare.com involving 25,000 school children and 17,854 health complaints submitted by parents. Researchers

found that masks are harming children physically, psychologically, and behaviourally with 24 health issues associated with mask wearing. They include: shortness of breath (29.7%); dizziness (26.4%); increased headaches (53%); difficulty concentrating (50%); drowsiness or fatigue (37%); and malaise (42%). Nearly a third of children experienced more sleep issues than before and a quarter developed new fears. Researchers found health issues and other impairments in 68 percent of masked children covering their faces for an average of 4.5 hours a day. Hundreds of those taking part experienced accelerated respiration, tightness in the chest, weakness, and short-term impairment of consciousness. A reminder of what Michie said again:

> The benefits of getting primary school children to wear masks is that regardless of what little degree of transmission is occurring in those age groups it could help normalise the practice. Young children wearing masks may be more likely to get their families to accept masks.

Psychopaths in government and psychology now have children and young people – plus all the adults – wearing masks for hours on end while clueless teachers impose the will of the psychopaths on the young they should be protecting. What the hell are parents doing?

Cult lab rats

We have some schools already imposing on students microchipped buzzers that activate when they get 'too close' to their pals in the way they do with lab rats. How apt. To the Cult and its brain-dead servants our children *are* lab rats being conditioned to be unquestioning, dehumanised slaves for the rest of their lives. Children and young people are being weaned and frightened away from the most natural human instincts including closeness and touch. I have tracked in the books over the years how schools were banning pupils from greeting each other with a hug and the whole Cult-induced Me Too movement has terrified men and boys from a relaxed and natural interaction with female friends and work colleagues to the point where many men try never to be in a room

alone with a woman that's not their partner. Airhead celebrities have as always played their virtue-signalling part in making this happen with their gross exaggeration. For every monster like Harvey Weinstein there are at least tens of thousands of men that don't treat women like that; but everyone must be branded the same and policy changed for them as well as the monster. I am going to be using the word 'dehumanise' many times in this chapter because that is what the Cult is seeking to do and it goes very deep as we shall see. Don't let them kid you that social distancing is planned to end one day. That's not the idea. We are seeing more governments and companies funding and producing wearable gadgets to keep people apart and they would not be doing that if this was meant to be short-term. A tech start-up company backed by GCHQ, the British Intelligence and military surveillance headquarters, has created a social distancing wrist sensor that alerts people when they get too close to others. The CIA has also supported tech companies developing similar devices. The wearable sensor was developed by Tended, one of a number of start-up companies supported by GCHQ (see the CIA and DARPA). The device can be worn on the wrist or as a tag on the waistband and will vibrate whenever someone wearing the device breaches social distancing and gets anywhere near natural human contact. The company had a lucky break in that it was developing a distancing sensor when the 'Covid' hoax arrived which immediately provided a potentially enormous market. How fortunate. The government in big-time Cult-controlled Ontario in Canada is investing $2.5 million in wearable contact tracing technology that 'will alert users if they may have been exposed to the Covid-19 in the workplace and will beep or vibrate if they are within six feet of another person'. Facedrive Inc., the technology company behind this, was founded in 2016 with funding from the Ontario Together Fund and obviously they, too, had a prophet on the board of directors. The human surveillance and control technology is called TraceSCAN and would be worn by the human cyborgs in places such as airports, workplaces, construction sites, care homes and … *schools*.

I emphasise schools with children and young people the prime targets. You know what is planned for society as a whole if you keep your eyes on the schools. They have always been places where the state program the next generation of slaves to be its compliant worker-ants – or Woker-ants these days; but in the mist of the 'Covid' madness they have been transformed into mind laboratories on a scale never seen before. Teachers and head teachers are just as programmed as the kids – often more so. Children are kept apart from human interaction by walk lanes, classroom distancing, staggered meal times, masks, and the rolling-out of buzzer systems. Schools are now physically laid out as a laboratory maze for lab-rats. Lunatics at a school in Anchorage, Alaska, who should be prosecuted for child abuse, took away desks and forced children to kneel (know your place) on a mat for five hours a day while wearing a mask and using their chairs as a desk. How this was supposed to impact on a 'virus' only these clinically insane people can tell you and even then it would be clap-trap. The school banned recess (interaction), art classes (creativity), and physical exercise (getting body and mind moving out of inertia). Everyone behind this outrage should be in jail or better still a mental institution. The behavioural manipulators are all for this dystopian approach to schools. Professor Susan Michie, the mind-doctor and British Communist Party member, said it was wrong to say that schools were safe. They had to be made so by 'distancing', masks and ventilation (sitting all day in the cold). I must ask this lady round for dinner on a night I know I am going to be out and not back for weeks. She probably wouldn't be able to make it, anyway, with all the visits to her own psychologist she must have block-booked.

Masking identity

I know how shocking it must be for you that a behaviour manipulator like Michie wants everyone to wear masks which have long been a feature of mind-control programs like the infamous MKUltra in the United States, but, there we are. We live and learn. I spent many years from 1996 to right across the millennium

researching mind control in detail on both sides of the Atlantic and elsewhere. I met a large number of mind-control survivors and many had been held captive in body and mind by MKUltra. MK stands for mind-control, but employs the German spelling in deference to the Nazis spirited out of Germany at the end of World War Two by Operation Paperclip in which the US authorities, with help from the Vatican, transported Nazi mind-controllers and engineers to America to continue their work. Many of them were behind the creation of NASA and they included Nazi scientist and SS officer Wernher von Braun who swapped designing V-2 rockets to bombard London with designing the Saturn V rockets that powered the NASA moon programme's Apollo craft. I think I may have mentioned that the Cult has no borders. Among Paperclip escapees was Josef Mengele, the Angel of Death in the Nazi concentration camps where he conducted mind and genetic experiments on children often using twins to provide a control twin to measure the impact of his 'work' on the other. If you want to observe the Cult mentality in all its extremes of evil then look into the life of Mengele. I have met many people who suffered mercilessly under Mengele in the United States where he operated under the name Dr Greene and became a stalwart of MKUltra programming and torture. Among his locations was the underground facility in the Mojave Desert in California called the China Lake Naval Weapons Station which is almost entirely below the surface. My books *The Biggest Secret*, *Children of the Matrix* and *The Perception Deception* have the detailed background to MKUltra.

The best-known MKUltra survivor is American Cathy O'Brien. I first met her and her late partner Mark Phillips at a conference in Colorado in 1996. Mark helped her escape and deprogram from decades of captivity in an offshoot of MKUltra known as Project Monarch in which 'sex slaves' were provided for the rich and famous including Father George Bush, Dick Cheney and the Clintons. Read Cathy and Mark's book *Trance-Formation of America* and if you are new to this you will be shocked to the core. I read it in 1996 shortly before, with the usual synchronicity of my life, I found

myself given a book table at the conference right next to hers. MKUltra never ended despite being very publicly exposed (only a small part of it) in the 1970s and continues in other guises. I am still in touch with Cathy. She contacted me during 2020 after masks became compulsory in many countries to tell me how they were used as part of MKUltra programming. I had been observing 'Covid regulations' and the relationship between authority and public for months. I saw techniques that I knew were employed on individuals in MKUltra being used on the global population. I had read many books and manuals on mind control including one called *Silent Weapons for Quiet Wars* which came to light in the 1980s and was a guide on how to perceptually program on a mass scale. 'Silent Weapons' refers to mind-control. I remembered a line from the manual as governments, medical authorities and law enforcement agencies have so obviously talked to – or rather at – the adult population since the 'Covid' hoax began as if they are children. The document said:

> If a person is spoken to by a T.V. advertiser as if he were a twelve-year-old, then, due to suggestibility, he will, with a certain probability, respond or react to that suggestion with the uncritical response of a twelve-year-old and will reach in to his economic reservoir and deliver its energy to buy that product on impulse when he passes it in the store.

That's why authority has spoken to adults like children since all this began.

Why did Michael Jackson wear masks?

Every aspect of the 'Covid' narrative has mind-control as its central theme. Cathy O'Brien wrote an article for davidicke.com about the connection between masks and mind control. Her daughter Kelly who I first met in the 1990s was born while Cathy was still held captive in MKUltra. Kelly was forced to wear a mask as part of her programming from the age of *two* to dehumanise her, target her sense of individuality and reduce the amount of oxygen her brain and body received. *Bingo*. This is the real reason for compulsory

masks, why they have been enforced en masse, and why they seek to increase the number they demand you wear. First one, then two, with one disgraceful alleged 'doctor' recommending four which is nothing less than a death sentence. Where and how often they must be worn is being expanded for the purpose of mass mind control and damaging respiratory health which they can call 'Covid-19'. Canada's government headed by the man-child Justin Trudeau, says it's fine for children of two and older to wear masks. An insane 'study' in Italy involving just 47 children concluded there was no problem for babies as young as *four months* wearing them. Even after people were 'vaccinated' they were still told to wear masks by the criminal that is Anthony Fauci. Cathy wrote that mandating masks is allowing the authorities literally to control the air we breathe which is what was done in MKUltra. You might recall how the singer Michael Jackson wore masks and there is a reason for that. He was subjected to MKUltra mind control through Project Monarch and his psyche was scrambled by these simpletons. Cathy wrote:

> In MKUltra Project Monarch mind control, Michael Jackson had to wear a mask to silence his voice so he could not reach out for help. Remember how he developed that whisper voice when he wasn't singing? Masks control the mind from the outside in, like the redefining of words is doing. By controlling what we can and cannot say for fear of being labeled racist or beaten, for example, it ultimately controls thought that drives our words and ultimately actions (or lack thereof).
>
> Likewise, a mask muffles our speech so that we are not heard, which controls voice … words … mind. This is Mind Control. Masks are an obvious mind control device, and I am disturbed so many people are complying on a global scale. Masks depersonalize while making a person feel as though they have no voice. It is a barrier to others. People who would never choose to comply but are forced to wear a mask in order to keep their job, and ultimately their family fed, are compromised. They often feel shame and are subdued. People have stopped talking with each other while media controls the narrative.

The 'no voice' theme has often become literal with train passengers told not to speak to each other in case they pass on the 'virus', singing banned for the same reason and bonkers California officials telling people riding roller coasters that they cannot shout and scream. Cathy said she heard every day from healed MKUltra survivors who cannot wear a mask without flashing back on ways

their breathing was controlled – 'from ball gags and penises to water boarding'. She said that through the years when she saw images of people in China wearing masks 'due to pollution' that it was really to control their oxygen levels. 'I knew it was as much of a population control mechanism of depersonalisation as are burkas', she said. Masks are another Chinese communist/fascist method of control that has been swept across the West as the West becomes China at lightning speed since we entered 2020.

Mask-19

There are other reasons for mandatory masks and these include destroying respiratory health to call it 'Covid-19' and stunting brain development of children and the young. Dr Margarite Griesz-Brisson MD, PhD, is a Consultant Neurologist and Neurophysiologist and the Founder and Medical Director of the London Neurology and Pain Clinic. Her CV goes down the street and round the corner. She is clearly someone who cares about people and won't parrot the propaganda. Griesz-Brisson has a PhD in pharmacology, with special interest in neurotoxicology, environmental medicine, neuroregeneration and neuroplasticity (the way the brain can change in the light of information received). She went public in October, 2020, with a passionate warning about the effects of mask-wearing laws:

> The reinhalation of our exhaled air will without a doubt create oxygen deficiency and a flooding of carbon dioxide. We know that the human brain is very sensitive to oxygen deprivation. There are nerve cells for example in the hippocampus that can't be longer than 3 minutes without oxygen – they cannot survive. The acute warning symptoms are headaches, drowsiness, dizziness, issues in concentration, slowing down of reaction time – reactions of the cognitive system.

Oh, I know, let's tell bus, truck and taxi drivers to wear them and people working machinery. How about pilots, doctors and police? Griesz-Brisson makes the important point that while the symptoms she mentions may fade as the body readjusts this does not alter the fact that people continue to operate in oxygen deficit with long list of

potential consequences. She said it was well known that neurodegenerative diseases take years or decades to develop. 'If today you forget your phone number, the breakdown in your brain would have already started 20 or 30 years ago.' She said degenerative processes in your brain are getting amplified as your oxygen deprivation continues through wearing a mask. Nerve cells in the brain are unable to divide themselves normally in these circumstances and lost nerve cells will no longer be regenerated. 'What is gone is gone.' Now consider that people like shop workers and *schoolchildren* are wearing masks for hours every day. What in the name of sanity is going to be happening to them? 'I do not wear a mask, I need my brain to think', Griesz-Brisson said, 'I want to have a clear head when I deal with my patients and not be in a carbon dioxide-induced anaesthesia'. If you are told to wear a mask anywhere ask the organisation, police, store, whatever, for their risk assessment on the dangers and negative effects on mind and body of enforcing mask-wearing. They won't have one because it has never been done not even by government. All of them must be subject to class-action lawsuits as the consequences come to light. They don't do mask risk assessments for an obvious reason. They know what the conclusions would be and independent scientific studies that *have* been done tell a horror story of consequences.

'Masks are criminal'

Dr Griesz-Brisson said that for children and adolescents, masks are an absolute no-no. They had an extremely active and adaptive immune system and their brain was incredibly active with so much to learn. 'The child's brain, or the youth's brain, is thirsting for oxygen.' The more metabolically active an organ was, the more oxygen it required; and in children and adolescents every organ was metabolically active. Griesz-Brisson said that to deprive a child's or adolescent's brain of oxygen, or to restrict it in any way, was not only dangerous to their health, it was absolutely criminal. 'Oxygen deficiency inhibits the development of the brain, and the damage that has taken place as a result CANNOT be reversed.' Mind

manipulators of MKUltra put masks on two-year-olds they wanted to neurologically rewire and you can see why. Griesz-Brisson said a child needs the brain to learn and the brain needs oxygen to function. 'We don't need a clinical study for that. This is simple, indisputable physiology.' Consciously and purposely induced oxygen deficiency was an absolutely deliberate health hazard, and an absolute medical contraindication which means that 'this drug, this therapy, this method or measure should not be used, and is not allowed to be used'. To coerce an entire population to use an absolute medical contraindication by force, she said, there had to be definite and serious reasons and the reasons must be presented to competent interdisciplinary and independent bodies to be verified and authorised. She had this warning of the consequences that were coming if mask wearing continued:

> When, in ten years, dementia is going to increase exponentially, and the younger generations couldn't reach their god-given potential, it won't help to say 'we didn't need the masks'. I know how damaging oxygen deprivation is for the brain, cardiologists know how damaging it is for the heart, pulmonologists know how damaging it is for the lungs. Oxygen deprivation damages every single organ. Where are our health departments, our health insurance, our medical associations? It would have been their duty to be vehemently against the lockdown and to stop it and stop it from the very beginning.
>
> Why do the medical boards issue punishments to doctors who give people exemptions? Does the person or the doctor seriously have to prove that oxygen deprivation harms people? What kind of medicine are our doctors and medical associations representing? Who is responsible for this crime? The ones who want to enforce it? The ones who let it happen and play along, or the ones who don't prevent it?

All of the organisations and people she mentions there either answer directly to the Cult or do whatever hierarchical levels above them tell them to do. The outcome of both is the same. 'It's not about masks, it's not about viruses, it's certainly not about your health', Griesz-Brisson said. 'It is about much, much more. I am not participating. I am not afraid.' They were taking our air to breathe and there was no unfounded medical exemption from face masks. Oxygen deprivation was dangerous for every single brain. It had to be the free decision of every human being whether they want to

wear a mask that was absolutely ineffective to protect themselves from a virus. She ended by rightly identifying where the responsibility lies for all this:

> The imperative of the hour is personal responsibility. We are responsible for what we think, not the media. We are responsible for what we do, not our superiors. We are responsible for our health, not the World Health Organization. And we are responsible for what happens in our country, not the government.

Halle-bloody-lujah.

But surgeons wear masks, right?

Independent studies of mask-wearing have produced a long list of reports detailing mental, emotional and physical dangers. What a definition of insanity to see police officers imposing mask-wearing on the public which will cumulatively damage their health while the police themselves wear masks that will cumulatively damage *their* health. It's utter madness and both public and police do this because 'the government says so' – yes a government of brain-donor idiots like UK Health Secretary Matt Hancock reading the 'follow the science' scripts of psychopathic, lunatic psychologists. The response you get from Stockholm syndrome sufferers defending the very authorities that are destroying them and their families is that 'surgeons wear masks'. This is considered the game, set and match that they must work and don't cause oxygen deficit. Well, actually, scientific studies have shown that they *do* and oxygen levels are monitored in operating theatres to compensate. Surgeons wear masks to stop spittle and such like dropping into open wounds – not to stop 'viral particles' which are so miniscule they can only be seen through an electron microscope. Holes in the masks are significantly bigger than 'viral particles' and if you sneeze or cough they will breach the mask. I watched an incredibly disingenuous 'experiment' that claimed to prove that masks work in catching 'virus' material from the mouth and nose. They did this with a slow motion camera and the mask did block big stuff which stayed inside the mask and

against the face to be breathed in or cause infections on the face as we have seen with many children. 'Viral particles', however, would never have been picked up by the camera as they came through the mask when they are far too small to be seen. The 'experiment' was therefore disingenuous *and* useless.

Studies have concluded that wearing masks in operating theatres (and thus elsewhere) make no difference to preventing infection while the opposite is true with toxic shite building up in the mask and this had led to an explosion in tooth decay and gum disease dubbed by dentists 'mask mouth'. You might have seen the Internet video of a furious American doctor urging people to take off their masks after a four-year-old patient had been rushed to hospital the night before and nearly died with a lung infection that doctors sourced to mask wearing. A study in the journal *Cancer Discovery* found that inhalation of harmful microbes can contribute to advanced stage lung cancer in adults and long-term use of masks can help breed dangerous pathogens. Microbiologists have said frequent mask wearing creates a moist environment in which microbes can grow and proliferate before entering the lungs. The Canadian Agency for Drugs and Technologies in Health, or CADTH, a Canadian national organisation that provides research and analysis to healthcare decision-makers, said this as long ago as 2013 in a report entitled 'Use of Surgical Masks in the Operating Room: A Review of the Clinical Effectiveness and Guidelines'. It said:

- No evidence was found to support the use of surgical face masks to reduce the frequency of surgical site infections
- No evidence was found on the effectiveness of wearing surgical face masks to protect staff from infectious material in the operating room.
- Guidelines recommend the use of surgical face masks by staff in the operating room to protect both operating room staff and patients (despite the lack of evidence).

We were told that the world could go back to 'normal' with the arrival of the 'vaccines'. When they came, fraudulent as they are, the story changed as I knew that it would. We are in the midst of transforming 'normal', not going back to it. Mary Ramsay, head of immunisation at Public Health England, echoed the words of US criminal Anthony Fauci who said masks and other regulations must stay no matter if people are vaccinated. The Fauci idiot continued to wear two masks – different colours so both could be clearly seen – after he *claimed* to have been vaccinated. Senator Rand Paul told Fauci in one exchange that his double-masks were 'theatre' and he was right. It's all theatre. Mary Ramsay back-tracked on the vaccine-return-to-normal theme when she said the public may need to wear masks and social-distance for years despite the jabs. 'People have got used to those lower-level restrictions now, and [they] can live with them', she said telling us what the idea has been all along. 'The vaccine does not give you a pass, even if you have had it, you must continue to follow all the guidelines' said a Public Health England statement which reneged on what we had been told before and made having the 'vaccine' irrelevant to 'normality' even by the official story. Spain's fascist government trumped everyone by passing a law mandating the wearing of masks on the beach and even when swimming in the sea. The move would have devastated what's left of the Spanish tourist industry, posed potential breathing dangers to swimmers and had Northern European sunbathers walking around with their forehead brown and the rest of their face white as a sheet. The ruling was so crazy that it had to be retracted after pressure from public and tourist industry, but it confirmed where the Cult wants to go with masks and how clinically insane authority has become. The determination to make masks permanent and hide the serious dangers to body and mind can be seen in the censorship of scientist Professor Denis Rancourt by Bill Gates-funded academic publishing website ResearchGate over his papers exposing the dangers and uselessness of masks. Rancourt said:

ResearchGate today has permanently locked my account, which I have had since 2015. Their reasons graphically show the nature of their attack against democracy, and their corruption of

science … By their obscene non-logic, a scientific review of science articles reporting on harms caused by face masks has a 'potential to cause harm'. No criticism of the psychological device (face masks) is tolerated, if the said criticism shows potential to influence public policy.

This is what happens in a fascist world.

Where are the 'greens' (again)?

Other dangers of wearing masks especially regularly relate to the inhalation of minute plastic fibres into the lungs and the deluge of discarded masks in the environment and oceans. Estimates predicted that more than 1.5 billion disposable masks will end up in the world's oceans every year polluting the water with tons of plastic and endangering marine wildlife. Studies project that humans are using 129 billion face masks each month worldwide – about three million a minute. Most are disposable and made from plastic, non-biodegradable microfibers that break down into smaller plastic particles that become widespread in ecosystems. They are littering cities, clogging sewage channels and turning up in bodies of water. I have written in other books about the immense amounts of microplastics from endless sources now being absorbed into the body. Rolf Halden, director of the Arizona State University (ASU) Biodesign Center for Environmental Health Engineering, was the senior researcher in a 2020 study that analysed 47 human tissue samples and found microplastics in all of them. 'We have detected these chemicals of plastics in every single organ that we have investigated', he said. I wrote in *The Answer* about the world being deluged with microplastics. A study by the Worldwide Fund for Nature (WWF) found that people are consuming on average every week some 2,000 tiny pieces of plastic mostly through water and also through marine life and the air. Every year humans are ingesting enough microplastics to fill a heaped dinner plate and in a life-time of 79 years it is enough to fill two large waste bins. Marco Lambertini, WWF International director general said: 'Not only are plastics polluting our oceans and waterways and killing marine life – it's in all of us and we can't escape consuming plastics,' American

geologists found tiny plastic fibres, beads and shards in rainwater samples collected from the remote slopes of the Rocky Mountain National Park near Denver, Colorado. Their report was headed: 'It is raining plastic.' Rachel Adams, senior lecturer in Biomedical Science at Cardiff Metropolitan University, said that among health consequences are internal inflammation and immune responses to a 'foreign body'. She further pointed out that microplastics become carriers of toxins including mercury, pesticides and dioxins (a known cause of cancer and reproductive and developmental problems). These toxins accumulate in the fatty tissues once they enter the body through microplastics. Now this is being compounded massively by people putting plastic on their face and throwing it away.

Workers exposed to polypropylene plastic fibres known as 'flock' have developed 'flock worker's lung' from inhaling small pieces of the flock fibres which can damage lung tissue, reduce breathing capacity and exacerbate other respiratory problems. *Now* … commonly used surgical masks have three layers of melt-blown textiles made of … polypropylene. We have billions of people putting these microplastics against their mouth, nose and face for hours at a time day after day in the form of masks. How does anyone think that will work out? I mean – what could possibly go wrong? We posted a number of scientific studies on this at davidicke.com, but when I went back to them as I was writing this book the links to the science research website where they were hosted were dead. Anything that challenges the official narrative in any way is either censored or vilified. The official narrative is so unsupportable by the evidence that only deleting the truth can protect it. A study by Chinese scientists still survived – with the usual twist which it why it was still active, I guess. Yes, they found that virtually all the masks they tested increased the daily intake of microplastic fibres, but people should still wear them because the danger from the 'virus' was worse said the crazy 'team' from the Institute of Hydrobiology in Wuhan. Scientists first discovered microplastics in lung tissue of some patients who died of lung cancer

in the 1990s. Subsequent studies have confirmed the potential health damage with the plastic degrading slowly and remaining in the lungs to accumulate in volume. Wuhan researchers used a machine simulating human breathing to establish that masks shed up to nearly 4,000 microplastic fibres in a month with reused masks producing more. Scientists said some masks are laced with toxic chemicals and a variety of compounds seriously restricted for both health and environmental reasons. They include cobalt (used in blue dye) and formaldehyde known to cause watery eyes, burning sensations in the eyes, nose, and throat, plus coughing, wheezing and nausea. No – that must be 'Covid-19'.

Mask 'worms'

There is another and potentially even more sinister content of masks. Mostly new masks of different makes filmed under a microscope around the world have been found to contain strange black fibres or 'worms' that appear to move or 'crawl' by themselves and react to heat and water. The nearest I have seen to them are the self-replicating fibres that are pulled out through the skin of those suffering from Morgellons disease which has been connected to the phenomena of 'chemtrails' which I will bring into the story later on. Morgellons fibres continue to grow outside the body and have a form of artificial intelligence. Black 'worm' fibres in masks have that kind of feel to them and there is a nanotechnology technique called 'worm micelles' which carry and release drugs or anything else you want to deliver to the body. For sure the suppression of humanity by mind altering drugs is the Cult agenda big time and the more excuses they can find to gain access to the body the more opportunities there are to make that happen whether through 'vaccines' or masks pushed against the mouth and nose for hours on end.

So let us summarise the pros and cons of masks:

Against masks: Breathing in your own carbon dioxide; depriving the body and brain of sufficient oxygen; build-up of toxins in the mask that can be breathed into the lungs and cause rashes on the face and 'mask-mouth'; breathing microplastic fibres and toxic chemicals into the lungs; dehumanisation and deleting individualisation by literally making people faceless; destroying human emotional interaction through facial expression and deleting parental connection with their babies which look for guidance to their facial expression.

For masks: They don't protect you from a 'virus' that doesn't exist and even if it did 'viral' particles are so minute they are smaller than the holes in the mask.

Governments, police, supermarkets, businesses, transport companies, and all the rest who seek to impose masks have done no risk assessment on their consequences for health and psychology and are now open to group lawsuits when the impact becomes clear with a cumulative epidemic of respiratory and other disease. Authorities will try to exploit these effects and hide the real cause by dubbing them 'Covid-19'. Can you imagine setting out to force the population to wear health-destroying masks without doing any assessment of the risks? It is criminal and it is evil, but then how many people targeted in this way, who see their children told to wear them all day at school, have asked for a risk assessment? Billions can't be imposed upon by the few unless the billions allow it. Oh, yes, with just a tinge of irony, 85 percent of all masks made worldwide come from *China*.

Wash your hands in toxic shite

'Covid' rules include the use of toxic sanitisers and again the health consequences of constantly applying toxins to be absorbed through the skin is obvious to any level of Renegade Mind. America's Food and Drug Administration (FDA) said that sanitisers are drugs and issued a warning about 75 dangerous brands which contain

methanol used in antifreeze and can cause death, kidney damage and blindness. The FDA circulated the following warning even for those brands that it claims to be safe:

> Store hand sanitizer out of the reach of pets and children, and children should use it only with adult supervision. Do not drink hand sanitizer. This is particularly important for young children, especially toddlers, who may be attracted by the pleasant smell or brightly colored bottles of hand sanitizer.
>
> Drinking even a small amount of hand sanitizer can cause alcohol poisoning in children. (However, there is no need to be concerned if your children eat with or lick their hands after using hand sanitizer.) During this coronavirus pandemic, poison control centers have had an increase in calls about accidental ingestion of hand sanitizer, so it is important that adults monitor young children's use.
>
> Do not allow pets to swallow hand sanitizer. If you think your pet has eaten something potentially dangerous, call your veterinarian or a pet poison control center right away. Hand sanitizer is flammable and should be stored away from heat and flames. When using hand sanitizer, rub your hands until they feel completely dry before performing activities that may involve heat, sparks, static electricity, or open flames.

There you go, perfectly safe, then, and that's without even a mention of the toxins absorbed through the skin. Come on kids – sanitise your hands everywhere you go. It will save you from the 'virus'. Put all these elements together of the 'Covid' normal and see how much health and psychology is being cumulatively damaged, even devastated, to 'protect your health'. Makes sense, right? They are only imposing these things because they care, right? *Right?*

Submitting to insanity

Psychological reframing of the population goes very deep and is done in many less obvious ways. I hear people say how contradictory and crazy 'Covid' rules are and how they are ever changing. This is explained away by dismissing those involved as idiots. It is a big mistake. The Cult is delighted if its cold calculation is perceived as incompetence and idiocy when it is anything but. Oh, yes, there are idiots within the system – lots of them – but they are *administering* the Cult agenda, mostly unknowingly. They are not deciding and dictating it. The bulwark against tyranny is self-

respect, always has been, always will be. It is self-respect that has broken every tyranny in history. By its very nature self-respect will not bow to oppression and its perpetrators. There is so little self-respect that it's always the few that overturn dictators. Many may eventually follow, but the few with the iron spines (self-respect) kick it off and generate the momentum. The Cult targets self-respect in the knowledge that once this has gone only submission remains. Crazy, contradictory, ever-changing 'Covid' rules are systematically applied by psychologists to delete self-respect. They *want* you to see that the rules make no sense. It is one thing to decide to do something when *you* have made the choice based on evidence and logic. You still retain your self-respect. It is quite another when you can see what you are being told to do is insane, ridiculous and makes no sense, and *yet you still do it*. Your self-respect is extinguished and this has been happening as ever more obviously stupid and nonsensical things have been demanded and the great majority have complied even when they can see they are stupid and nonsensical.

People walk around in face-nappies knowing they are damaging their health and make no difference to a 'virus'. They do it in fear of not doing it. I know it's daft, but I'll do it anyway. When that happens something dies inside of you and submissive reframing has begun. Next there's a need to hide from yourself that you have conceded your self-respect and you convince yourself that you have not really submitted to fear and intimidation. You begin to believe that you are complying with craziness because it's the right thing to do. When first you concede your self-respect of 2+2 = 4 to 2+2 = 5 you *know* you are compromising your self-respect. Gradually to avoid facing that fact you begin to *believe* that 2+2=5. You have been reframed and I have been watching this process happening in the human psyche on an industrial scale. The Cult is working to break your spirit and one of its major tools in that war is humiliation. I read how former American soldier Bradley Manning (later Chelsea Manning after a sex-change) was treated after being jailed for supplying WikiLeaks with documents exposing the enormity of

government and elite mendacity. Manning was isolated in solitary confinement for eight months, put under 24-hour surveillance, forced to hand over clothing before going to bed, and stand naked for every roll call. This is systematic humiliation. The introduction of anal swab 'Covid' tests in China has been done for the same reason to delete self-respect and induce compliant submission. Anal swabs are mandatory for incoming passengers in parts of China and American diplomats have said they were forced to undergo the indignity which would have been calculated humiliation by the Cult-owned Chinese government that has America in its sights.

Government-people: An abusive relationship

Spirit-breaking psychological techniques include giving people hope and apparent respite from tyranny only to take it away again. This happened in the UK during Christmas, 2020, when the psycho-psychologists and their political lackeys announced an easing of restrictions over the holiday only to reimpose them almost immediately on the basis of yet another lie. There is a big psychological difference between getting used to oppression and being given hope of relief only to have that dashed. Psychologists know this and we have seen the technique used repeatedly. Then there is traumatising people before you introduce more extreme regulations that require compliance. A perfect case was the announcement by the dark and sinister Whitty and Vallance in the UK that 'new data' predicted that 4,000 could die every day over the winter of 2020/2021 if we did not lockdown again. I think they call it lying and after traumatising people with that claim out came Jackboot Johnson the next day with new curbs on human freedom. Psychologists know that a frightened and traumatised mind becomes suggestable to submission and behaviour reframing. Underpinning all this has been to make people fearful and suspicious of each other and see themselves as a potential danger to others. In league with deleted self-respect you have the perfect psychological recipe for self-loathing. The relationship between authority and public is now demonstrably the same as that of

subservience to an abusive partner. These are signs of an abusive relationship explained by psychologist Leslie Becker-Phelps:

Psychological and emotional abuse: Undermining a partner's self-worth with verbal attacks, name-calling, and belittling. Humiliating the partner in public, unjustly accusing them of having an affair, or interrogating them about their every behavior. Keeping partner confused or off balance by saying they were just kidding or blaming the partner for 'making' them act this way … Feigning in public that they care while turning against them in private. This leads to victims frequently feeling confused, incompetent, unworthy, hopeless, and chronically self-doubting. [Apply these techniques to how governments have treated the population since New Year, 2020, and the parallels are obvious.]

Physical abuse: The abuser might physically harm their partner in a range of ways, such as grabbing, hitting, punching, or shoving them. They might throw objects at them or harm them with a weapon. [Observe the physical harm imposed by masks, lockdown, and so on.]

Threats and intimidation: One way abusers keep their partners in line is by instilling fear. They might be verbally threatening, or give threatening looks or gestures. Abusers often make it known that they are tracking their partner's every move. They might destroy their partner's possessions, threaten to harm them, or threaten to harm their family members. Not surprisingly, victims of this abuse often feel anxiety, fear, and panic. [No words necessary.]

Isolation: Abusers often limit their partner's activities, forbidding them to talk or interact with friends or family. They might limit access to a car or even turn off their phone. All of this might be done by physically holding them against their will, but is often accomplished through psychological abuse and intimidation. The more isolated a person feels, the fewer resources they have to help gain perspective on their situation and to escape from it. [No words necessary.]

Economic abuse: Abusers often make their partners beholden to them for money by controlling access to funds of any kind. They might prevent their partner from getting a job or withhold access to money they earn from a job. This creates financial dependency that makes leaving the relationship very difficult. [See destruction of livelihoods and the proposed meagre 'guaranteed income' so long as you do whatever you are told.]

Using children: An abuser might disparage their partner's parenting skills, tell their children lies about their partner, threaten to take custody of their children, or threaten to harm their children. These tactics instil fear and often elicit compliance. [See reframed social service mafia and how children are being mercilessly abused by the state over 'Covid' while their parents look on too frightened to do anything.]

A further recurring trait in an abusive relationship is the abused blaming themselves for their abuse and making excuses for the abuser. We have the public blaming each other for lockdown abuse by government and many making excuses for the government while attacking those who challenge the government. How often we have heard authorities say that rules are being imposed or reimposed only because people have refused to 'behave' and follow the rules. We don't want to do it – it's *you*.

Renegade Minds are an antidote to all of these things. They will never concede their self-respect no matter what the circumstances. Even when apparent humiliation is heaped upon them they laugh in its face and reflect back the humiliation on the abuser where it belongs. Renegade Minds will never wear masks they know are only imposed to humiliate, suppress and damage both physically and psychologically. Consequences will take care of themselves and they will never break their spirit or cause them to concede to tyranny. UK newspaper columnist Peter Hitchens was one of the few in the mainstream media to speak out against lockdowns and forced vaccinations. He then announced he had taken the jab. He wanted to see family members abroad and he believed vaccine passports were inevitable even though they had not yet been introduced. Hitchens

has a questioning and critical mind, but not a Renegade one. If he had no amount of pressure would have made him concede. Hitchens excused his action by saying that the battle has been lost. Renegade Minds never accept defeat when freedom is at stake and even if they are the last one standing the self-respect of not submitting to tyranny is more important than any outcome or any consequence.

That's why Renegade Minds are the only minds that ever changed anything worth changing.

CHAPTER EIGHT

'Reframing' insanity

Insanity is relative. It depends on who has who locked in what cage
Ray Bradbury

'Reframing' a mind means simply to change its perception and behaviour. This can be done subconsciously to such an extent that subjects have no idea they have been 'reframed' while to any observer changes in behaviour and attitudes are obvious.

Human society is being reframed on a ginormous scale since the start of 2020 and here we have the reason why psychologists rather than doctors have been calling the shots. Ask most people who have succumbed to 'Covid' reframing if they have changed and most will say 'no'; but they *have* and fundamentally. The Cult's long-game has been preparing for these times since way back and crucial to that has been to prepare both population and officialdom mentally and emotionally. To use the mind-control parlance they had to reframe the population with a mentality that would submit to fascism and reframe those in government and law enforcement to impose fascism or at least go along with it. The result has been the fact-deleted mindlessness of 'Wokeness' and officialdom that has either enthusiastically or unquestioningly imposed global tyranny demanded by reframed politicians on behalf of psychopathic and deeply evil cultists. 'Cognitive reframing' identifies and challenges the way someone sees the world in the form of situations, experiences and emotions and then restructures those perceptions to view the same set of circumstances in a different way. This can have

benefits if the attitudes are personally destructive while on the other side it has the potential for individual and collective mind control which the subject has no idea has even happened.

Cognitive therapy was developed in the 1960s by Aaron T. Beck who was born in Rhode Island in 1921 as the son of Jewish immigrants from the Ukraine. He became interested in the techniques as a treatment for depression. Beck's daughter Judith S. Beck is prominent in the same field and they founded the Beck Institute for Cognitive Behavior Therapy in Philadelphia in 1994. Cognitive reframing, however, began to be used worldwide by those with a very dark agenda. The Cult reframes politicians to change their attitudes and actions until they are completely at odds with what they once appeared to stand for. The same has been happening to government administrators at all levels, law enforcement, military and the human population. Cultists love mind control for two main reasons: It allows them to control what people think, do and say to secure agenda advancement and, by definition, it calms their legendary insecurity and fear of the unexpected. I have studied mind control since the time I travelled America in 1996. I may have been talking to next to no one in terms of an audience in those years, but my goodness did I gather a phenomenal amount of information and knowledge about so many things including the techniques of mind control. I have described this in detail in other books going back to *The Biggest Secret* in 1998. I met a very large number of people recovering from MKUltra and its offshoots and successors and I began to see how these same techniques were being used on the population in general. This was never more obvious than since the 'Covid' hoax began.

Reframing the enforcers

I have observed over the last two decades and more the very clear transformation in the dynamic between the police, officialdom and the public. I tracked this in the books as the relationship mutated from one of serving the public to seeing them as almost the enemy and certainly a lower caste. There has always been a class divide

based on income and always been some psychopathic, corrupt, and big-I-am police officers. This was different. Wholesale change was unfolding in the collective dynamic; it was less about money and far more about position and perceived power. An us-and-them was emerging. Noses were lifted skyward by government administration and law enforcement and their attitude to the public they were *supposed* to be serving changed to one of increasing contempt, superiority and control. The transformation was so clear and widespread that it had to be planned. Collective attitudes and dynamics do not change naturally and organically that quickly on that scale. I then came across an organisation in Britain called Common Purpose created in the late 1980s by Julia Middleton who would work in the office of Deputy Prime Minister John Prescott during the long and disastrous premiership of war criminal Tony Blair. When Blair speaks the Cult is speaking and the man should have been in jail a long time ago. Common Purpose proclaims itself to be one of the biggest 'leadership development' organisations in the world while functioning as a *charity* with all the financial benefits which come from that. It hosts 'leadership development' courses and programmes all over the world and claims to have 'brought together' what it calls 'leaders' from more than 100 countries on six continents. The modus operandi of Common Purpose can be compared with the work of the UK government's reframing network that includes the Behavioural Insights Team 'nudge unit' and 'Covid' reframing specialists at SPI-B. WikiLeaks described Common Purpose long ago as 'a hidden virus in our government and schools' which is unknown to the general public: 'It recruits and trains "leaders" to be loyal to the directives of Common Purpose and the EU, instead of to their own departments, which they then undermine or subvert, the NHS [National Health Service] being an example.' This is a vital point to understand the 'Covid' hoax. The NHS, and its equivalent around the world, has been utterly reframed in terms of administrators and much of the medical personnel with the transformation underpinned by recruitment policies. The outcome has been the criminal and psychopathic behaviour of the

NHS over 'Covid' and we have seen the same in every other major country. WikiLeaks said Common Purpose trainees are 'learning to rule without regard to democracy' and to usher in a police state (current events explained). Common Purpose operated like a 'glue' and had members in the NHS, BBC, police, legal profession, church, many of Britain's 7,000 quangos, local councils, the Civil Service, government ministries and Parliament, and controlled many RDA's (Regional Development Agencies). Here we have one answer for how and why British institutions and their like in other countries have changed so negatively in relation to the public. This further explains how and why the beyond-disgraceful reframed BBC has become a propaganda arm of 'Covid' fascism. They are all part of a network pursuing the same goal.

By 2019 Common Purpose was quoting a figure of 85,000 'leaders' that had attended its programmes. These 'students' of all ages are known as Common Purpose 'graduates' and they consist of government, state and local government officials and administrators, police chiefs and officers, and a whole range of others operating within the national, local and global establishment. Cressida Dick, Commissioner of the London Metropolitan Police, is the Common Purpose graduate who was the 'Gold Commander' that oversaw what can only be described as the murder of Brazilian electrician Jean Charles de Menezes in 2005. He was held down by psychopathic police and shot seven times in the head by a psychopathic lunatic after being mistaken for a terrorist when he was just a bloke going about his day. Dick authorised officers to pursue and keep surveillance on de Menezes and ordered that he be stopped from entering the underground train system. Police psychopaths took her at her word clearly. She was 'disciplined' for this outrage by being *promoted* – eventually to the top of the 'Met' police where she has been a disaster. Many Chief Constables controlling the police in different parts of the UK are and have been Common Purpose graduates. I have heard the 'graduate' network described as a sort of Mafia or secret society operating within the fabric of government at all levels pursuing a collective policy

ingrained at Common Purpose training events. Founder Julia Middleton herself has said:

> Locally and internationally, Common Purpose graduates will be 'lighting small fires' to create change in their organisations and communities … The Common Purpose effect is best illustrated by the many stories of small changes brought about by leaders, who themselves have changed.

A Common Purpose mission statement declared:

> Common Purpose aims to improve the way society works by expanding the vision, decision-making ability and influence of all kinds of leaders. The organisation runs a variety of educational programmes for leaders of all ages, backgrounds and sectors, in order to provide them with the inspirational, information and opportunities they need to change the world.

Yes, but into what? Since 2020 the answer has become clear.

NLP and the Delphi technique

Common Purpose would seem to be a perfect name or would common programming be better? One of the foundation methods of reaching 'consensus' (group think) is by setting the agenda theme and then encouraging, cajoling or pressuring everyone to agree a 'consensus' in line with the core theme promoted by Common Purpose. The methodology involves the 'Delphi technique', or an adaption of it, in which opinions are expressed that are summarised by a 'facilitator or change agent' at each stage. Participants are 'encouraged' to modify their views in the light of what others have said. Stage by stage the former individual opinions are merged into group consensus which just happens to be what Common Purpose wants them to believe. A key part of this is to marginalise anyone refusing to concede to group think and turn the group against them to apply pressure to conform. We are seeing this very technique used on the general population to make 'Covid' group-thinkers hostile to those who have seen through the bullshit. People can be reframed by using perception manipulation methods such as Neuro-Linguistic Programming (NLP) in which you change perception with the use of

carefully constructed language. An NLP website described the technique this way:

> … A method of influencing brain behaviour (the 'neuro' part of the phrase) through the use of language (the 'linguistic' part) and other types of communication to enable a person to 'recode' the way the brain responds to stimuli (that's the 'programming') and manifest new and better behaviours. Neuro-Linguistic Programming often incorporates hypnosis and self-hypnosis to help achieve the change (or 'programming') that is wanted.

British alternative media operation UKColumn has done very detailed research into Common Purpose over a long period. I quoted co-founder and former naval officer Brian Gerrish in my book *Remember Who You Are*, published in 2011, as saying the following years before current times:

> It is interesting that many of the mothers who have had children taken by the State speak of the Social Services people being icily cool, emotionless and, as two ladies said in slightly different words, '… like little robots'. We know that NLP is cumulative, so people can be given small imperceptible doses of NLP in a course here, another in a few months, next year etc. In this way, major changes are accrued in their personality, but the day by day change is almost unnoticeable.

In these and other ways 'graduates' have had their perceptions uniformly reframed and they return to their roles in the institutions of government, law enforcement, legal profession, military, 'education', the UK National Health Service and the whole swathe of the establishment structure to pursue a common agenda preparing for the 'post-industrial', 'post-democratic' society. I say 'preparing' but we are now there. 'Post-industrial' is code for the Great Reset and 'post-democratic' is 'Covid' fascism. UKColumn has spoken to partners of those who have attended Common Purpose 'training'. They have described how personalities and attitudes of 'graduates' changed very noticeably for the worse by the time they had completed the course. They had been 'reframed' and told they are the 'leaders' – the special ones – who know better than the population. There has also been the very demonstrable recruitment of psychopaths and narcissists into government administration at all

levels and law enforcement. If you want psychopathy hire psychopaths and you get a simple cause and effect. If you want administrators, police officers and 'leaders' to perceive the public as lesser beings who don't matter then employ narcissists. These personalities are identified using 'psychometrics' that identifies knowledge, abilities, attitudes and personality traits, mostly through carefully-designed questionnaires and tests. As this policy has passed through the decades we have had power-crazy, power-trippers appointed into law enforcement, security and government administration in preparation for current times and the dynamic between public and law enforcement/officialdom has been transformed. UKColumn's Brian Gerrish said of the narcissistic personality:

> Their love of themselves and power automatically means that they will crush others who get in their way. I received a major piece of the puzzle when a friend pointed out that when they made public officials re-apply for their own jobs several years ago they were also required to do psychometric tests. This was undoubtedly the start of the screening process to get 'their' sort of people in post.

How obvious that has been since 2020 although it was clear what was happening long before if people paid attention to the changing public-establishment dynamic.

Change agents

At the centre of events in 'Covid' Britain is the National Health Service (NHS) which has behaved disgracefully in slavishly following the Cult agenda. The NHS management structure is awash with Common Purpose graduates or 'change agents' working to a common cause. Helen Bevan, a Chief of Service Transformation at the NHS Institute for Innovation and Improvement, co-authored a document called 'Towards a million change agents, a review of the social movements literature: implications for large scale change in the NHS'. The document compared a project management approach to that of change and social movements where 'people change

themselves and each other – peer to peer'. Two definitions given for a 'social movement' were:

A group of people who consciously attempt to build a radically new social order; involves people of a broad range of social backgrounds; and deploys politically confrontational and socially disruptive tactics – Cyrus Zirakzadeh 1997

Collective challenges, based on common purposes and social solidarities, in sustained interaction with elites, opponents, and authorities – Sidney Tarrow 1994

Helen Bevan wrote another NHS document in which she defined 'framing' as 'the process by which leaders construct, articulate and put across their message in a powerful and compelling way in order to win people to their cause and call them to action'. I think I could come up with another definition that would be rather more accurate. The National Health Service and institutions of Britain and the wider world have been taken over by reframed 'change agents' and that includes everything from the United Nations to national governments, local councils and social services which have been kidnapping children from loving parents on an extraordinary and gathering scale on the road to the end of parenthood altogether. Children from loving homes are stolen and kidnapped by the state and put into the 'care' (inversion) of the local authority through council homes, foster parents and forced adoption. At the same time children are allowed to be abused without response while many are under council 'care'. UKColumn highlighted the Common Purpose connection between South Yorkshire Police and Rotherham council officers in the case of the scandal in that area of the sexual exploitation of children to which the authorities turned not one blind eye, but both:

We were alarmed to discover that the Chief Executive, the Strategic Director of Children and Young People's Services, the Manager for the Local Strategic Partnership, the Community Cohesion Manager, the Cabinet Member for Cohesion, the Chief Constable and his predecessor had all attended Leadership training courses provided by the pseudo-charity Common Purpose.

Once 'change agents' have secured positions of hire and fire within any organisation things start to move very quickly. Personnel are then hired and fired on the basis of whether they will work towards the agenda the change agent represents. If they do they are rapidly promoted even though they may be incompetent. Those more qualified and skilled who are pre-Common Purpose 'old school' see their careers stall and even disappear. This has been happening for decades in every institution of state, police, 'health' and social services and all of them have been transformed as a result in their attitudes to their jobs and the public. Medical professions, including nursing, which were once vocations for the caring now employ many cold, callous and couldn't give a shit personality types. The UKColumn investigation concluded:

By blurring the boundaries between people, professions, public and private sectors, responsibility and accountability, Common Purpose encourages 'graduates' to believe that as new selected leaders, they can work together, outside of the established political and social structures, to achieve a paradigm shift or CHANGE – so called 'Leading Beyond Authority'. In doing so, the allegiance of the individual becomes 'reframed' on CP colleagues and their NETWORK.

Reframing the Face-Nappies

Nowhere has this process been more obvious than in the police where recruitment of psychopaths and development of unquestioning mind-controlled group-thinkers have transformed law enforcement into a politically-correct 'Woke' joke and a travesty of what should be public service. Today they wear their face-nappies like good little gofers and enforce 'Covid' rules which are fascism under another name. Alongside the specifically-recruited psychopaths we have software minds incapable of free thought. Brian Gerrish again:

An example is the policeman who would not get on a bike for a press photo because he had not done the cycling proficiency course. Normal people say this is political correctness gone mad. Nothing could be further from the truth. The policeman has been reframed, and in his reality it is perfect common sense not to get on the bike 'because he hasn't done the cycling course'.

Another example of this is where the police would not rescue a boy from a pond until they had taken advice from above on the 'risk assessment'. A normal person would have arrived, perhaps thought of the risk for a moment, and dived in. To the police now 'reframed', they followed 'normal' procedure.

There are shocking cases of reframed ambulance crews doing the same. Sheer unthinking stupidity of London Face-Nappies headed by Common Purpose graduate Cressida Dick can be seen in their behaviour at a vigil in March, 2021, for a murdered woman, Sarah Everard. A police officer had been charged with the crime. Anyone with a brain would have left the vigil alone in the circumstances. Instead they 'manhandled' women to stop them breaking 'Covid rules' to betray classic reframing. Minds in the thrall of perception control have no capacity for seeing a situation on its merits and acting accordingly. 'Rules is rules' is their only mind-set. My father used to say that rules and regulations are for the guidance of the intelligent and the blind obedience of the idiot. Most of the intelligent, decent, coppers have gone leaving only the other kind and a few old school for whom the job must be a daily nightmare. The combination of psychopaths and rule-book software minds has been clearly on public display in the 'Covid' era with automaton robots in uniform imposing fascistic 'Covid' regulations on the population without any personal initiative or judging situations on their merits. There are thousands of examples around the world, but I'll make my point with the infamous Derbyshire police in the English East Midlands – the ones who think pouring dye into beauty spots and using drones to track people walking in the countryside away from anyone is called 'policing'. To them there are rules decreed by the government which they have to enforce and in their bewildered state a group gathering in a closed space and someone walking alone in the countryside are the same thing. It is beyond idiocy and enters the realm of clinical insanity.

Police officers in Derbyshire said they were 'horrified' – *horrified* – to find 15 to 20 'irresponsible' kids playing a football match at a closed leisure centre 'in breach of coronavirus restrictions'. When they saw the police the kids ran away leaving their belongings behind and the reframed men and women of Derbyshire police were seeking to establish their identities with a view to fining their parents. The most natural thing for youngsters to do – kicking a ball about – is turned into a criminal activity and enforced by the moronic software programs of Derbyshire police. You find the same mentality in every country. These barely conscious 'horrified' officers said they had to take action because 'we need to ensure these rules are being followed' and 'it is of the utmost importance that you ensure your children are following the rules and regulations for Covid-19'. Had any of them done ten seconds of research to see if this parroting of their masters' script could be supported by any evidence? Nope. Reframed people don't think – others think for them and that's the whole idea of reframing. I have seen police officers one after the other repeating without question word for word what officialdom tells them just as I have seen great swathes of the public doing the same. Ask either for 'their' opinion and out spews what they have been told to think by the official narrative. Police and public may seem to be in different groups, but their mentality is the same. Most people do whatever they are told in fear not doing so or because they believe what officialdom tells them; almost the entirety of the police do what they are told for the same reason. Ultimately it's the tiny inner core of the global Cult that's telling both what to do.

So Derbyshire police were 'horrified'. Oh, really? Why did they think those kids were playing football? It was to relieve the psychological consequences of lockdown and being denied human contact with their friends and interaction, touch and discourse vital to human psychological health. Being denied this month after month has dismantled the psyche of many children and young people as depression and suicide have exploded. Were Derbyshire police *horrified by that*? Are you kidding? Reframed people don't have those

mental and emotional processes that can see how the impact on the psychological health of youngsters is far more dangerous than any 'virus' even if you take the mendacious official figures to be true. The reframed are told (programmed) how to act and so they do. The Derbyshire Chief Constable in the first period of lockdown when the black dye and drones nonsense was going on was Peter Goodman. He was the man who severed the connection between his force and the Derbyshire Constabulary *Male Voice* Choir when he decided that it was not inclusive enough to allow women to join. The fact it was a male voice choir making a particular sound produced by male voices seemed to elude a guy who terrifyingly ran policing in Derbyshire. He retired weeks after his force was condemned as disgraceful by former Supreme Court Justice Jonathan Sumption for their behaviour over extreme lockdown impositions. Goodman was replaced by his deputy Rachel Swann who was in charge when her officers were 'horrified'. The police statement over the boys committing the hanging-offence of playing football included the line about the youngsters being 'irresponsible in the times we are all living through' missing the point that the real relevance of the 'times we are all living through' is the imposition of fascism enforced by psychopaths and reframed minds of police officers playing such a vital part in establishing the fascist tyranny that their own children and grandchildren will have to live in their entire lives. As a definition of insanity that is hard to beat although it might be run close by imposing masks on people that can have a serious effect on their health while wearing a face nappy all day themselves. Once again public and police do it for the same reason – the authorities tell them to and who are they to have the self-respect to say no?

Wokers in uniform

How reframed do you have to be to arrest a *six-year-old* and take him to court for *picking a flower* while waiting for a bus? Brain dead police and officialdom did just that in North Carolina where criminal proceedings happen regularly for children under nine. Attorney Julie Boyer gave the six-year-old crayons and a colouring book

during the 'flower' hearing while the 'adults' decided his fate. County Chief District Court Judge Jay Corpening asked: 'Should a child that believes in Santa Claus, the Easter Bunny and the tooth fairy be making life-altering decisions?' Well, of course not, but common sense has no meaning when you have a common purpose and a reframed mind. Treating children in this way, and police operating in American schools, is all part of the psychological preparation for children to accept a police state as normal all their adult lives. The same goes for all the cameras and biometric tracking technology in schools. Police training is focused on reframing them as snowflake Wokers and this is happening in the military. Pentagon top brass said that 'training sessions on extremism' were needed for troops who asked why they were so focused on the Capitol Building riot when Black Lives Matter riots were ignored. What's the difference between them some apparently and rightly asked. Actually, there is a difference. Five people died in the Capitol riot, only one through violence, and that was a police officer shooting an unarmed protestor. BLM riots killed at least 25 people and cost billions. Asking the question prompted the psychopaths and reframed minds that run the Pentagon to say that more 'education' (programming) was needed. Troop training is all based on psychological programming to make them fodder for the Cult – 'Military men are just dumb, stupid animals to be used as pawns in foreign policy' as Cult-to-his-DNA former Secretary of State Henry Kissinger famously said. Governments see the police in similar terms and it's time for those among them who can see this to defend the people and stop being enforcers of the Cult agenda upon the people.

The US military, like the country itself, is being targeted for destruction through a long list of Woke impositions. Cult-owned gaga 'President' Biden signed an executive order when he took office to allow taxpayer money to pay for transgender surgery for active military personnel and veterans. Are you a man soldier? No, I'm a LGBTQIA+ with a hint of Skoliosexual and Spectrasexual. Oh, good man. Bad choice of words you bigot. The Pentagon announced in March, 2021, the appointment of the first 'diversity and inclusion

officer' for US Special Forces. Richard Torres-Estrada arrived with the publication of a 'D&I Strategic Plan which will guide the enterprise-wide effort to institutionalize and sustain D&I'. If you think a Special Forces 'Strategic Plan' should have something to do with defending America you haven't been paying attention. Defending Woke is now the military's new role. Torres-Estrada has posted images comparing Donald Trump with Adolf Hitler and we can expect no bias from him as a representative of the supposedly non-political Pentagon. Cable news host Tucker Carlson said: 'The Pentagon is now the Yale faculty lounge but with cruise missiles.' Meanwhile Secretary of Defense Lloyd Austin, a board member of weapons-maker Raytheon with stock and compensation interests in October, 2020, worth $1.4 million, said he was purging the military of the 'enemy within' – anyone who isn't Woke and supports Donald Trump. Austin refers to his targets as 'racist extremists' while in true Woke fashion being himself a racist extremist. Pentagon documents pledge to 'eradicate, eliminate and conquer all forms of racism, sexism and homophobia'. The definitions of these are decided by 'diversity and inclusion committees' peopled by those who see racism, sexism and homophobia in every situation and opinion. Woke (the Cult) is dismantling the US military and purging testosterone as China expands its military and gives its troops 'masculinity training'. How do we think that is going to end when this is all Cult coordinated? The US military, like the British military, is controlled by Woke and spineless top brass who just go along with it out of personal career interests.

'Woke' means fast asleep

Mind control and perception manipulation techniques used on individuals to create group-think have been unleashed on the global population in general. As a result many have no capacity to see the obvious fascist agenda being installed all around them or what 'Covid' is really all about. Their brains are firewalled like a computer system not to process certain concepts, thoughts and realisations that are bad for the Cult. The young are most targeted as the adults they

will be when the whole fascist global state is planned to be fully implemented. They need to be prepared for total compliance to eliminate all pushback from entire generations. The Cult has been pouring billions into taking complete control of 'education' from schools to universities via its operatives and corporations and not least Bill Gates as always. The plan has been to transform 'education' institutions into programming centres for the mentality of 'Woke'. James McConnell, professor of psychology at the University of Michigan, wrote in *Psychology Today* in 1970:

> The day has come when we can combine sensory deprivation with drugs, hypnosis, and astute manipulation of reward and punishment, to gain almost absolute control over an individual's behaviour. It should then be possible to achieve a very rapid and highly effective type of brainwashing that would allow us to make dramatic changes in a person's behaviour and personality ...
>
> … We should reshape society so that we all would be trained from birth to want to do what society wants us to do. We have the techniques to do it... no-one owns his own personality you acquired, and there's no reason to believe you should have the right to refuse to acquire a new personality if your old one is anti-social.

This was the potential for mass brainwashing in 1970 and the mentality there displayed captures the arrogant psychopathy that drives it forward. I emphasise that not all young people have succumbed to Woke programming and those that haven't are incredibly impressive people given that today's young are the most perceptually-targeted generations in history with all the technology now involved. Vast swathes of the young generations, however, have fallen into the spell – and that's what it is – of Woke. The Woke mentality and perceptual program is founded on *inversion* and you will appreciate later why that is so significant. Everything with Woke is inverted and the opposite of what it is claimed to be. Woke was a term used in African-American culture from the 1900s and referred to an awareness of social and racial justice. This is not the meaning of the modern version or 'New Woke' as I call it in *The Answer*. Oh, no, Woke today means something very different no matter how much Wokers may seek to hide that and insist Old Woke and New

Woke are the same. See if you find any 'awareness of social justice' here in the modern variety:

- Woke demands 'inclusivity' while excluding anyone with a different opinion and calls for mass censorship to silence other views.

- Woke claims to stand against oppression when imposing oppression is the foundation of all that it does. It is the driver of political correctness which is nothing more than a Cult invention to manipulate the population to silence itself.

- Woke believes itself to be 'liberal' while pursuing a global society that can only be described as fascist (see 'anti-fascist' fascist Antifa).

- Woke calls for 'social justice' while spreading injustice wherever it goes against the common 'enemy' which can be easily identified as a differing view.

- Woke is supposed to be a metaphor for 'awake' when it is solid-gold asleep and deep in a Cult-induced coma that meets the criteria for 'off with the fairies'.

I state these points as obvious facts if people only care to look. I don't do this with a sense of condemnation. We need to appreciate that the onslaught of perceptual programming on the young has been incessant and merciless. I can understand why so many have been reframed, or, given their youth, framed from the start to see the world as the Cult demands. The Cult has had access to their minds day after day in its 'education' system for their entire formative years. Perception is formed from information received and the Cult-created system is a life-long download of information delivered to elicit a particular perception, thus behaviour. The more this has expanded into still new extremes in recent decades and ever-increasing censorship has deleted other opinions and information why wouldn't that lead to a perceptual reframing on a mass scale? I

have described already cradle-to-grave programming and in more recent times the targeting of young minds from birth to adulthood has entered the stratosphere. This has taken the form of skewing what is 'taught' to fit the Cult agenda and the omnipresent techniques of group-think to isolate non-believers and pressure them into line. There has always been a tendency to follow the herd, but we really are in a new world now in relation to that. We have parents who can see the 'Covid' hoax told by their children not to stop them wearing masks at school, being 'Covid' tested or having the 'vaccine' in fear of the peer-pressure consequences of being different. What is 'peer-pressure' if not pressure to conform to group-think? Renegade Minds never group-think and always retain a set of perceptions that are unique to them. Group-think is always underpinned by consequences for not group-thinking. Abuse now aimed at those refusing DNA-manipulating 'Covid vaccines' are a potent example of this. The biggest pressure to conform comes from the very group which is itself being manipulated. 'I am programmed to be part of a hive mind and so you must be.'

Woke control structures in 'education' now apply to every mainstream organisation. Those at the top of the 'education' hierarchy (the Cult) decide the policy. This is imposed on governments through the Cult network; governments impose it on schools, colleges and universities; their leadership impose the policy on teachers and academics and they impose it on children and students. At any level where there is resistance, perhaps from a teacher or university lecturer, they are targeted by the authorities and often fired. Students themselves regularly demand the dismissal of academics (increasingly few) at odds with the narrative that the students have been programmed to believe in. It is quite a thought that students who are being targeted by the Cult become so consumed by programmed group-think that they launch protests and demand the removal of those who are trying to push back against those targeting the students. Such is the scale of perceptual inversion. We see this with 'Covid' programming as the Cult imposes the rules via psycho-psychologists and governments on

shops, transport companies and businesses which impose them on their staff who impose them on their customers who pressure Pushbackers to conform to the will of the Cult which is in the process of destroying them and their families. Scan all aspects of society and you will see the same sequence every time.

Fact free Woke and hijacking the 'left'

There is no more potent example of this than 'Woke', a mentality only made possible by the deletion of factual evidence by an 'education' system seeking to produce an ever more uniform society. Why would you bother with facts when you don't know any? Deletion of credible history both in volume and type is highly relevant. Orwell said: 'Who controls the past controls the future: who controls the present controls the past.' They who control the perception of the past control the perception of the future and they who control the present control the perception of the past through the writing and deleting of history. Why would you oppose the imposition of Marxism in the name of Wokeism when you don't know that Marxism cost at least 100 million lives in the 20th century alone? Watch videos and read reports in which Woker generations are asked basic historical questions – it's mind-blowing. A survey of 2,000 people found that six percent of millennials (born approximately early1980s to early 2000s) believed the Second World War (1939-1945) broke out with the assassination of President Kennedy (in 1963) and one in ten thought Margaret Thatcher was British Prime Minister at the time. She was in office between 1979 and 1990. We are in a post-fact society. Provable facts are no defence against the fascism of political correctness or Silicon Valley censorship. Facts don't matter anymore as we have witnessed with the 'Covid' hoax. Sacrificing uniqueness to the Woke group-think religion is all you are required to do and that means thinking for yourself is the biggest Woke no, no. All religions are an expression of group-think and censorship and Woke is just another religion with an orthodoxy defended by group-think and censorship. Burned at

the stake becomes burned on Twitter which leads back eventually to burned at the stake as Woke humanity regresses to ages past.

The biggest Woke inversion of all is its creators and funders. I grew up in a traditional left of centre political household on a council estate in Leicester in the 1950s and 60s – you know, the left that challenged the power of wealth-hoarding elites and threats to freedom of speech and opinion. In those days students went on marches defending freedom of speech while today's Wokers march for its deletion. What on earth could have happened? Those very elites (collectively the Cult) that we opposed in my youth and early life have funded into existence the antithesis of that former left and hijacked the 'brand' while inverting everything it ever stood for. We have a mentality that calls itself 'liberal' and 'progressive' while acting like fascists. Cult billionaires and their corporations have funded themselves into control of 'education' to ensure that Woke programming is unceasing throughout the formative years of children and young people and that non-Wokers are isolated (that word again) whether they be students, teachers or college professors. The Cult has funded into existence the now colossal global network of Woke organisations that have spawned and promoted all the 'causes' on the Cult wish-list for global transformation and turned Wokers into demanders of them. Does anyone really think it's a coincidence that the Cult agenda for humanity is a carbon (sorry) copy of the societal transformations desired by Woke?? These are only some of them:

Political correctness: The means by which the Cult deletes all public debates that it knows it cannot win if we had the free-flow of information and evidence.

Human-caused 'climate change': The means by which the Cult seeks to transform society into a globally-controlled dictatorship imposing its will over the fine detail of everyone's lives 'to save the planet' which doesn't actually need saving.

Transgender obsession: Preparing collective perception to accept the 'new human' which would not have genders because it would be created technologically and not through procreation. I'll have much more on this in Human 2.0.

Race obsession: The means by which the Cult seeks to divide and rule the population by triggering racial division through the perception that society is more racist than ever when the opposite is the case. Is it perfect in that regard? No. But to compare today with the racism of apartheid and segregation brought to an end by the civil rights movement in the 1960s is to insult the memory of that movement and inspirations like Martin Luther King. Why is the 'anti-racism' industry (which it is) so dominated by privileged white people?

White supremacy: This is a label used by privileged white people to demonise poor and deprived white people pushing back on tyranny to marginalise and destroy them. White people are being especially targeted as the dominant race by number within Western society which the Cult seeks to transform in its image. If you want to change a society you must weaken and undermine its biggest group and once you have done that by using the other groups you next turn on them to do the same … 'Then they came for the Jews and I was not a Jew so I did nothing.'

Mass migration: The mass movement of people from the Middle East, Africa and Asia into Europe, from the south into the United States and from Asia into Australia are another way the Cult seeks to dilute the racial, cultural and political influence of white people on Western society. White people ask why their governments appear to be working against them while being politically and culturally biased towards incoming cultures. Well, here's your answer. In the same way sexually 'straight' people, men and women, ask why the

authorities are biased against them in favour of other sexualities. The answer is the same – that's the way the Cult wants it to be for very sinister motives.

These are all central parts of the Cult agenda and central parts of the Woke agenda and Woke was created and continues to be funded to an immense degree by Cult billionaires and corporations. If anyone begins to say 'coincidence' the syllables should stick in their throat.

Billionaire 'social justice warriors'

Joe Biden is a 100 percent-owned asset of the Cult and the Wokers' man in the White House whenever he can remember his name and for however long he lasts with his rapidly diminishing cognitive function. Even walking up the steps of an aircraft without falling on his arse would appear to be a challenge. He's not an empty-shell puppet or anything. From the minute Biden took office (or the Cult did) he began his executive orders promoting the Woke wish-list. You will see the Woke agenda imposed ever more severely because it's really the *Cult* agenda. Woke organisations and activist networks spawned by the Cult are funded to the extreme so long as they promote what the Cult wants to happen. Woke is funded to promote 'social justice' by billionaires who become billionaires by destroying social justice. The social justice mantra is only a cover for dismantling social justice and funded by billionaires that couldn't give a damn about social justice. Everything makes sense when you see that. One of Woke's premier funders is Cult billionaire financier George Soros who said: 'I am basically there to make money, I cannot and do not look at the social consequences of what I do.' This is the same Soros who has given more than $32 billion to his Open Society Foundations global Woke network and funded Black Lives Matter, mass immigration into Europe and the United States, transgender activism, climate change activism, political correctness and groups targeting 'white supremacy' in the form of privileged white thugs that dominate Antifa. What a scam it all is and when

you are dealing with the unquestioning fact-free zone of Woke scamming them is child's play. All you need to pull it off in all these organisations are a few in-the-know agents of the Cult and an army of naïve, reframed, uninformed, narcissistic, know-nothings convinced of their own self-righteousness, self-purity and virtue.

Soros and fellow billionaires and billionaire corporations have poured hundreds of millions into Black Lives Matter and connected groups and promoted them to a global audience. None of this is motivated by caring about black people. These are the billionaires that have controlled and exploited a system that leaves millions of black people in abject poverty and deprivation which they do absolutely nothing to address. The same Cult networks funding BLM were behind the *slave trade!* Black Lives Matter hijacked a phrase that few would challenge and they have turned this laudable concept into a political weapon to divide society. You know that BLM is a fraud when it claims that *All* Lives Matter, the most inclusive statement of all, is 'racist'. BLM and its Cult masters don't want to end racism. To them it's a means to an end to control all of humanity never mind the colour, creed, culture or background. What has destroying the nuclear family got to do with ending racism? Nothing – but that is one of the goals of BLM and also happens to be a goal of the Cult as I have been exposing in my books for decades. Stealing children from loving parents and giving schools ever more power to override parents is part of that same agenda. BLM is a Marxist organisation and why would that not be the case when the Cult created Marxism *and* BLM? Patrisse Cullors, a BLM co-founder, said in a 2015 video that she and her fellow organisers, including co-founder Alicia Garza, are 'trained Marxists'. The lady known after marriage as Patrisse Khan-Cullors bought a $1.4 million home in 2021 in one of the whitest areas of California with a black population of just 1.6 per cent and has so far bought *four* high-end homes for a total of $3.2 million. How very Marxist. There must be a bit of spare in the BLM coffers, however, when Cult corporations and billionaires have handed over the best part of $100 million. Many black people can see that Black Lives Matter is not

working for them, but against them, and this is still more confirmation. Black journalist Jason Whitlock, who had his account suspended by Twitter for simply linking to the story about the 'Marxist's' home buying spree, said that BLM leaders are 'making millions of dollars off the backs of these dead black men who they wouldn't spit on if they were on fire and alive'.

Black Lies Matter

Cult assets and agencies came together to promote BLM in the wake of the death of career criminal George Floyd who had been jailed a number of times including for forcing his way into the home of a black woman with others in a raid in which a gun was pointed at her stomach. Floyd was filmed being held in a Minneapolis street in 2020 with the knee of a police officer on his neck and he subsequently died. It was an appalling thing for the officer to do, but the same technique has been used by police on peaceful protestors of lockdown without any outcry from the Woke brigade. As unquestioning supporters of the Cult agenda Wokers have supported lockdown and all the 'Covid' claptrap while attacking anyone standing up to the tyranny imposed in its name. Court documents would later include details of an autopsy on Floyd by County Medical Examiner Dr Andrew Baker who concluded that Floyd had taken a fatal level of the drug fentanyl. None of this mattered to fact-free, question-free, Woke. Floyd's death was followed by worldwide protests against police brutality amid calls to defund the police. Throwing babies out with the bathwater is a Woke speciality. In the wake of the murder of British woman Sarah Everard a Green Party member of the House of Lords, Baroness Jones of Moulescoomb (Nincompoopia would have been better), called for a 6pm curfew for all men. This would be in breach of the Geneva Conventions on war crimes which ban collective punishment, but that would never have crossed the black and white Woke mind of Baroness Nincompoopia who would have been far too convinced of her own self-righteousness to compute such details. Many American cities did defund the police in the face of Floyd riots

and after $15 million was deleted from the police budget in Washington DC under useless Woke mayor Muriel Bowser car-jacking alone rose by 300 percent and within six months the US capital recorded its highest murder rate in 15 years. The same happened in Chicago and other cities in line with the Cult/Soros plan to bring fear to streets and neighbourhoods by reducing the police, releasing violent criminals and not prosecuting crime. This is the mob-rule agenda that I have warned in the books was coming for so long. Shootings in the area of Minneapolis where Floyd was arrested increased by 2,500 percent compared with the year before. Defunding the police over George Floyd has led to a big increase in dead people with many of them black. Police protection for politicians making these decisions stayed the same or increased as you would expect from professional hypocrites. The Cult doesn't actually want to abolish the police. It wants to abolish local control over the police and hand it to federal government as the psychopaths advance the Hunger Games Society. Many George Floyd protests turned into violent riots with black stores and businesses destroyed by fire and looting across America fuelled by Black Lives Matter. Woke doesn't do irony. If you want civil rights you must loot the liquor store and the supermarket and make off with a smart TV. It's the only way.

It's not a race war – it's a class war

Black people are patronised by privileged blacks and whites alike and told they are victims of white supremacy. I find it extraordinary to watch privileged blacks supporting the very system and bloodline networks behind the slave trade and parroting the same Cult-serving manipulative crap of their privileged white, often billionaire, associates. It is indeed not a race war but a class war and colour is just a diversion. Black Senator Cory Booker and black Congresswoman Maxine Waters, more residents of Nincompoopia, personify this. Once you tell people they are victims of someone else you devalue both their own responsibility for their plight and the power they have to impact on their reality and experience. Instead

we have: 'You are only in your situation because of whitey – turn on them and everything will change.' It won't change. Nothing changes in our lives unless *we* change it. Crucial to that is never seeing yourself as a victim and always as the creator of your reality. Life is a simple sequence of choice and consequence. Make different choices and you create different consequences. *You* have to make those choices – not Black Lives Matter, the Woke Mafia and anyone else that seeks to dictate your life. Who are they these Wokers, an emotional and psychological road traffic accident, to tell you what to do? Personal empowerment is the last thing the Cult and its Black Lives Matter want black people or anyone else to have. They claim to be defending the underdog while *creating* and perpetuating the underdog. The Cult's worst nightmare is human unity and if they are going to keep blacks, whites and every other race under economic servitude and control then the focus must be diverted from what they have in common to what they can be manipulated to believe divides them. Blacks have to be told that their poverty and plight is the fault of the white bloke living on the street in the same poverty and with the same plight they are experiencing. The difference is that your plight black people is due to him, a white supremacist with 'white privilege' living on the street. Don't unite as one human family against your mutual oppressors and suppressors – fight the oppressor with the white face who is as financially deprived as you are. The Cult knows that as its 'Covid' agenda moves into still new levels of extremism people are going to respond and it has been spreading the seeds of disunity everywhere to stop a united response to the evil that targets *all of us*.

Racist attacks on 'whiteness' are getting ever more outrageous and especially through the American Democratic Party which has an appalling history for anti-black racism. Barack Obama, Joe Biden, Hillary Clinton and Nancy Pelosi all eulogised about Senator Robert Byrd at his funeral in 2010 after a nearly 60-year career in Congress. Byrd was a brutal Ku Klux Klan racist and a violent abuser of Cathy O'Brien in MKUltra. He said he would never fight in the military 'with a negro by my side' and 'rather I should die a thousand times,

and see Old Glory trampled in the dirt never to rise again, than to see this beloved land of ours become degraded by race mongrels, a throwback to the blackest specimen from the wilds'. Biden called Byrd a 'very close friend and mentor'. These 'Woke' hypocrites are not anti-racist they are anti-poor and anti-people not of their perceived class. Here is an illustration of the scale of anti-white racism to which we have now descended. Seriously Woke and moronic *New York Times* contributor Damon Young described whiteness as a 'virus' that 'like other viruses will not die until there are no bodies left for it to infect'. He went on: '… the only way to stop it is to locate it, isolate it, extract it, and kill it.' Young can say that as a black man with no consequences when a white man saying the same in reverse would be facing a jail sentence. *That's* racism. We had super-Woke numbskull senators Tammy Duckworth and Mazie Hirono saying they would object to future Biden Cabinet appointments if he did not nominate more Asian Americans and Pacific Islanders. Never mind the ability of the candidate what do they look like? Duckworth said: 'I will vote for racial minorities and I will vote for LGBTQ, but anyone else I'm not voting for.' Appointing people on the grounds of race is illegal, but that was not a problem for this ludicrous pair. They were on-message and that's a free pass in any situation.

Critical race racism

White children are told at school they are intrinsically racist as they are taught the divisive 'critical race theory'. This claims that the law and legal institutions are inherently racist and that race is a socially constructed concept used by white people to further their economic and political interests at the expense of people of colour. White is a 'virus' as we've seen. Racial inequality results from 'social, economic, and legal differences that white people create between races to maintain white interests which leads to poverty and criminality in minority communities'. I must tell that to the white guy sleeping on the street. The principal of East Side Community School in New York sent white parents a manifesto that called on

them to become 'white traitors' and advocate for full 'white abolition'. These people are teaching your kids when they urgently need a psychiatrist. The 'school' included a chart with 'eight white identities' that ranged from 'white supremacist' to 'white abolition' and defined the behaviour white people must follow to end 'the regime of whiteness'. Woke blacks and their privileged white associates are acting exactly like the slave owners of old and Ku Klux Klan racists like Robert Byrd. They are too full of their own self-purity to see that, but it's true. Racism is not a body type; it's a state of mind that can manifest through any colour, creed or culture.

Another racial fraud is *'equity'*. Not equality of treatment and opportunity – equity. It's a term spun as equality when it means something very different. Equality in its true sense is a raising up while 'equity' is a race to the bottom. Everyone in the same level of poverty is 'equity'. Keep everyone down – that's equity. The Cult doesn't want anyone in the human family to be empowered and BLM leaders, like all these 'anti-racist' organisations, continue their privileged, pampered existence by perpetuating the perception of gathering racism. When is the last time you heard an 'anti-racist' or 'anti-Semitism' organisation say that acts of racism and discrimination have *fallen?* It's not in the interests of their fund-raising and power to influence and the same goes for the professional soccer anti-racism operation, Kick It Out. Two things confirmed that the Black Lives Matter riots in the summer of 2020 were Cult creations. One was that while anti-lockdown protests were condemned in this same period for 'transmitting 'Covid' the authorities supported mass gatherings of Black Lives Matter supporters. I even saw self-deluding people claiming to be doctors say the two types of protest were not the same. No – the non-existent 'Covid' was in favour of lockdowns and attacked those that protested against them while 'Covid' supported Black Lives Matter and kept well away from its protests. The whole thing was a joke and as lockdown protestors were arrested, often brutally, by reframed Face-Nappies we had the grotesque sight of police officers taking the knee to Black Lives Matter, a Cult-funded Marxist

organisation that supports violent riots and wants to destroy the nuclear family and white people.

He's not white? Shucks!

Woke obsession with race was on display again when ten people were shot dead in Boulder, Colorado, in March, 2021. Cult-owned Woke TV channels like CNN said the shooter appeared to be a white man and Wokers were on Twitter condemning 'violent white men' with the usual mantras. Then the shooter's name was released as Ahmad Al Aliwi Alissa, an anti-Trump Arab-American, and the sigh of disappointment could be heard five miles away. Never mind that ten people were dead and what that meant for their families. Race baiting was all that mattered to these sick Cult-serving people like Barack Obama who exploited the deaths to further divide America on racial grounds which is his job for the Cult. This is the man that 'racist' white Americans made the first black president of the United States and then gave him a second term. Not-very-bright Obama has become filthy rich on the back of that and today appears to have a big influence on the Biden administration. Even so he's still a downtrodden black man and a victim of white supremacy. This disingenuous fraud reveals the contempt he has for black people when he puts on a Deep South Alabama accent whenever he talks to them, no, *at* them.

Another BLM red flag was how the now fully-Woke (fully-Cult) and fully-virtue-signalled professional soccer authorities had their teams taking the knee before every match in support of Marxist Black Lives Matter. Soccer authorities and clubs displayed 'Black Lives Matter' on the players' shirts and flashed the name on electronic billboards around the pitch. Any fans that condemned what is a Freemasonic taking-the-knee ritual were widely condemned as you would expect from the Woke virtue-signallers of professional sport and the now fully-Woke media. We have reverse racism in which you are banned from criticising any race or culture except for white people for whom anything goes – say what you like, no problem. What has this got to do with racial harmony and

equality? We've had black supremacists from Black Lives Matter telling white people to fall to their knees in the street and apologise for their white supremacy. Black supremacists acting like white supremacist slave owners of the past couldn't breach their self-obsessed, race-obsessed sense of self-purity. Joe Biden appointed a race-obsessed black supremacist Kristen Clarke to head the Justice Department Civil Rights Division. Clarke claimed that blacks are endowed with 'greater mental, physical and spiritual abilities' than whites. If anyone reversed that statement they would be vilified. Clarke is on-message so no problem. She's never seen a black-white situation in which the black figure is anything but a virtuous victim and she heads the Civil Rights Division which should treat everyone the same or it isn't civil rights. Another perception of the Renegade Mind: If something or someone is part of the Cult agenda they will be supported by Woke governments and media no matter what. If they're not, they will be condemned and censored. It really is that simple and so racist Clarke prospers despite (make that because of) her racism.

The end of culture

Biden's administration is full of such racial, cultural and economic bias as the Cult requires the human family to be divided into warring factions. We are now seeing racially-segregated graduations and everything, but everything, is defined through the lens of perceived 'racism. We have 'racist' mathematics, 'racist' food and even 'racist' *plants*. World famous Kew Gardens in London said it was changing labels on plants and flowers to tell its pre-'Covid' more than two million visitors a year how racist they are. Kew director Richard Deverell said this was part of an effort to 'move quickly to decolonise collections' after they were approached by one Ajay Chhabra 'an actor with an insight into how sugar cane was linked to slavery'. They are *plants* you idiots. 'Decolonisation' in the Woke manual really means colonisation of society with its mentality and by extension colonisation by the Cult. We are witnessing a new Chinese-style 'Cultural Revolution' so essential to the success of all

Marxist takeovers. Our cultural past and traditions have to be swept away to allow a new culture to be built-back-better. Woke targeting of long-standing Western cultural pillars including historical monuments and cancelling of historical figures is what happened in the Mao revolution in China which 'purged remnants of capitalist and traditional elements from Chinese society' and installed Maoism as the dominant ideology'. For China see the Western world today and for 'dominant ideology' see Woke. Better still see Marxism or Maoism. The 'Covid' hoax has specifically sought to destroy the arts and all elements of Western culture from people meeting in a pub or restaurant to closing theatres, music venues, sports stadiums, places of worship and even banning *singing*. Destruction of Western society is also why criticism of any religion is banned except for Christianity which again is the dominant religion as white is the numerically-dominant race. Christianity may be fading rapidly, but its history and traditions are weaved through the fabric of Western society. Delete the pillars and other structures will follow until the whole thing collapses. I am not a Christian defending that religion when I say that. I have no religion. It's just a fact. To this end Christianity has itself been turned Woke to usher its own downfall and its ranks are awash with 'change agents' – knowing and unknowing – at every level including Pope Francis (*definitely* knowing) and the clueless Archbishop of Canterbury Justin Welby (possibly not, but who can be sure?). Woke seeks to coordinate attacks on Western culture, traditions, and ways of life through 'intersectionality' defined as 'the complex, cumulative way in which the effects of multiple forms of discrimination (such as racism, sexism, and classism) combine, overlap, or intersect especially in the experiences of marginalised individuals or groups'. Wade through the Orwellian Woke-speak and this means coordinating disparate groups in a common cause to overthrow freedom and liberal values.

The entire structure of public institutions has been infested with Woke – government at all levels, political parties, police, military, schools, universities, advertising, media and trade unions. This abomination has been achieved through the Cult web by appointing

Wokers to positions of power and battering non-Wokers into line through intimidation, isolation and threats to their job. Many have been fired in the wake of the empathy-deleted, vicious hostility of 'social justice' Wokers and the desire of gutless, spineless employers to virtue-signal their Wokeness. Corporations are filled with Wokers today, most notably those in Silicon Valley. Ironically at the top they are not Woke at all. They are only exploiting the mentality their Cult masters have created and funded to censor and enslave while the Wokers cheer them on until it's their turn. Thus the Woke 'liberal left' is an inversion of the traditional liberal left. Campaigning for justice on the grounds of power and wealth distribution has been replaced by campaigning for identity politics. The genuine traditional left would never have taken money from today's billionaire abusers of fairness and justice and nor would the billionaires have wanted to fund that genuine left. It would not have been in their interests to do so. The division of opinion in those days was between the haves and have nots. This all changed with Cult manipulated and funded identity politics. The division of opinion today is between Wokers and non-Wokers and not income brackets. Cult corporations and their billionaires may have taken wealth disparity to cataclysmic levels of injustice, but as long as they speak the language of Woke, hand out the dosh to the Woke network and censor the enemy they are 'one of us'. Billionaires who don't give a damn about injustice are laughing at them till their bellies hurt. Wokers are not even close to self-aware enough to see that. The transformed 'left' dynamic means that Wokers who drone on about 'social justice' are funded by billionaires that have destroyed social justice the world over. It's *why* they are billionaires.

The climate con

Nothing encapsulates what I have said more comprehensively than the hoax of human-caused global warming. I have detailed in my books over the years how Cult operatives and organisations were the pump-primers from the start of the climate con. A purpose-built vehicle for this is the Club of Rome established by the Cult in 1968

with the Rockefellers and Rothschilds centrally involved all along. Their gofer frontman Maurice Strong, a Canadian oil millionaire, hosted the Earth Summit in Rio de Janeiro, Brazil, in 1992 where the global 'green movement' really expanded in earnest under the guiding hand of the Cult. The Earth Summit established Agenda 21 through the Cult-created-and-owned United Nations to use the illusion of human-caused climate change to justify the transformation of global society to save the world from climate disaster. It is a No-Problem-Reaction-Solution sold through governments, media, schools and universities as whole generations have been terrified into believing that the world was going to end in their lifetimes unless what old people had inflicted upon them was stopped by a complete restructuring of how everything is done. Chill, kids, it's all a hoax. Such restructuring is precisely what the Cult agenda demands (purely by coincidence of course). Today this has been given the codename of the Great Reset which is only an updated term for Agenda 21 and its associated Agenda 2030. The latter, too, is administered through the UN and was voted into being by the General Assembly in 2015. Both 21 and 2030 seek centralised control of all resources and food right down to the raindrops falling on your own land. These are some of the demands of Agenda 21 established in 1992. See if you recognise this society emerging today:

- End national sovereignty
- State planning and management of all land resources, ecosystems, deserts, forests, mountains, oceans and fresh water; agriculture; rural development; biotechnology; and ensuring *'equity'*
- The state to 'define the role' of business and financial resources
- Abolition of private property
- 'Restructuring' the family unit (see BLM)
- Children raised by the state
- People told what their job will be
- Major restrictions on movement
- Creation of 'human settlement zones'

- Mass resettlement as people are forced to vacate land where they live
- Dumbing down education
- Mass global depopulation in pursuit of all the above

The United Nations was created as a Trojan horse for world government. With the climate con of critical importance to promoting that outcome you would expect the UN to be involved. Oh, it's involved all right. The UN is promoting Agenda 21 and Agenda 2030 justified by 'climate change' while also driving the climate hoax through its Intergovernmental Panel on Climate Change (IPCC), one of the world's most corrupt organisations. The IPCC has been lying ferociously and constantly since the day it opened its doors with the global media hanging unquestioningly on its every mendacious word. The Green movement is entirely Woke and has long lost its original environmental focus since it was co-opted by the Cult. An obsession with 'global warming' has deleted its values and scrambled its head. I experienced a small example of what I mean on a beautiful country walk that I have enjoyed several times a week for many years. The path merged into the fields and forests and you felt at one with the natural world. Then a 'Green' organisation, the Hampshire and Isle of Wight Wildlife Trust, took over part of the land and proceeded to cut down a large number of trees, including mature ones, to install a horrible big, bright steel 'this-is-ours-stay-out' fence that destroyed the whole atmosphere of this beautiful place. No one with a feel for nature would do that. Day after day I walked to the sound of chainsaws and a magnificent mature weeping willow tree that I so admired was cut down at the base of the trunk. When I challenged a Woke young girl in a green shirt (of course) about this vandalism she replied: 'It's a weeping willow – it will grow back.' This is what people are paying for when they donate to the Hampshire and Isle of Wight Wildlife Trust and many other 'green' organisations today. It is not the environmental movement that I knew and instead has become a support-system – as with Extinction Rebellion – for a very dark agenda.

Private jets for climate justice

The Cult-owned, Gates-funded, World Economic Forum and its founder Klaus Schwab were behind the emergence of Greta Thunberg to harness the young behind the climate agenda and she was invited to speak to the world at … the UN. Schwab published a book, *Covid-19: The Great Reset* in 2020 in which he used the 'Covid' hoax and the climate hoax to lay out a new society straight out of Agenda 21 and Agenda 2030. Bill Gates followed in early 2021 when he took time out from destroying the world to produce a book in his name about the way to save it. Gates flies across the world in private jets and admitted that 'I probably have one of the highest greenhouse gas footprints of anyone on the planet … my personal flying alone is gigantic.' He has also bid for the planet's biggest private jet operator. Other climate change saviours who fly in private jets include John Kerry, the US Special Presidential Envoy for Climate, and actor Leonardo DiCaprio, a 'UN Messenger of Peace with special focus on climate change'. These people are so full of bullshit they could corner the market in manure. We mustn't be sceptical, though, because the Gates book, *How to Avoid a Climate Disaster: The Solutions We Have and the Breakthroughs We Need*, is a genuine attempt to protect the world and not an obvious pile of excrement attributed to a mega-psychopath aimed at selling his masters' plans for humanity. The Gates book and the other shite-pile by Klaus Schwab could have been written by the same person and may well have been. Both use 'climate change' and 'Covid' as the excuses for their new society and by coincidence the Cult's World Economic Forum and Bill and Melinda Gates Foundation promote the climate hoax and hosted Event 201 which pre-empted with a 'simulation' the very 'coronavirus' hoax that would be simulated for real on humanity within weeks. The British 'royal' family is promoting the 'Reset' as you would expect through Prince 'climate change caused the war in Syria' Charles and his hapless son Prince William who said that we must 'reset our relationship with nature and our trajectory as a species' to avoid a climate disaster. Amazing how many promotors of the 'Covid' and 'climate change' control

systems are connected to Gates and the World Economic Forum. A 'study' in early 2021 claimed that carbon dioxide emissions must fall by the equivalent of a global lockdown roughly every two years for the next decade to save the planet. The 'study' appeared in the same period that the Schwab mob claimed in a video that lockdowns destroying the lives of billions are good because they make the earth 'quieter' with less 'ambient noise'. They took down the video amid a public backlash for such arrogant, empathy-deleted stupidity You see, however, where they are going with this. Corinne Le Quéré, a professor at the Tyndall Centre for Climate Change Research, University of East Anglia, was lead author of the climate lockdown study, and she writes for … the World Economic Forum. Gates calls in 'his' book for changing 'every aspect of the economy' (long-time Cult agenda) and for humans to eat synthetic 'meat' (predicted in my books) while cows and other farm animals are eliminated. Australian TV host and commentator Alan Jones described what carbon emission targets would mean for farm animals in Australia alone if emissions were reduced as demanded by 35 percent by 2030 and zero by 2050:

> Well, let's take agriculture, the total emissions from agriculture are about 75 million tonnes of carbon dioxide, equivalent. Now reduce that by 35 percent and you have to come down to 50 million tonnes, I've done the maths. So if you take for example 1.5 million cows, you're going to have to reduce the herd by 525,000 [by] 2030, nine years, that's 58,000 cows a year. The beef herd's 30 million, reduce that by 35 percent, that's 10.5 million, which means 1.2 million cattle have to go every year between now and 2030. This is insanity!
>
> There are 75 million sheep. Reduce that by 35 percent, that's 26 million sheep, that's almost 3 million a year. So under the Paris Agreement over 30 million beasts. dairy cows, cattle, pigs and sheep would go. More than 8,000 every minute of every hour for the next decade, do these people know what they're talking about?

Clearly they don't at the level of campaigners, politicians and administrators. The Cult *does* know; that's the outcome it wants. We are faced with not just a war on humanity. Animals and the natural world are being targeted and I have been saying since the 'Covid' hoax began that the plan eventually was to claim that the 'deadly virus' is able to jump from animals, including farm animals and

domestic pets, to humans. Just before this book went into production came this story: 'Russia registers world's first Covid-19 vaccine for cats & dogs as makers of Sputnik V warn pets & farm animals could spread virus'. The report said 'top scientists warned that the deadly pathogen could soon begin spreading through homes and farms' and 'the next stage is the infection of farm and domestic animals'. Know the outcome and you'll see the journey. Think what that would mean for animals and keep your eye on a term called zoonosis or zoonotic diseases which transmit between animals and humans. The Cult wants to break the connection between animals and people as it does between people and people. Farm animals fit with the Cult agenda to transform food from natural to synthetic.

The gas of life is killing us

There can be few greater examples of Cult inversion than the condemnation of carbon dioxide as a dangerous pollutant when it is the gas of life. Without it the natural world would be dead and so we would all be dead. We breathe in oxygen and breathe out carbon dioxide while plants produce oxygen and absorb carbon dioxide. It is a perfect symbiotic relationship that the Cult wants to dismantle for reasons I will come to in the final two chapters. Gates, Schwab, other Cult operatives and mindless repeaters, want the world to be 'carbon neutral' by at least 2050 and the earlier the better. 'Zero carbon' is the cry echoed by lunatics calling for 'Zero Covid' when we already have it. These carbon emission targets will deindustrialise the world in accordance with Cult plans – the post-industrial, post-democratic society – and with so-called renewables like solar and wind not coming even close to meeting human energy needs blackouts and cold are inevitable. Texans got the picture in the winter of 2021 when a snow storm stopped wind turbines and solar panels from working and the lights went down along with water which relies on electricity for its supply system. Gates wants everything to be powered by electricity to ensure that his masters have the kill switch to stop all human activity, movement, cooking, water and warmth any time they like. The climate lie is so

stupendously inverted that it claims we must urgently reduce carbon dioxide when we *don't have enough*.

Co2 in the atmosphere is a little above 400 parts per million when the optimum for plant growth is 2,000 ppm and when it falls anywhere near 150 ppm the natural world starts to die and so do we. It fell to as low as 280 ppm in an 1880 measurement in Hawaii and rose to 413 ppm in 2019 with industrialisation which is why the planet has become *greener* in the industrial period. How insane then that psychopathic madman Gates is not satisfied only with blocking the rise of Co2. He's funding technology to suck it out of the atmosphere. The reason why will become clear. The industrial era is not destroying the world through Co2 and has instead turned around a potentially disastrous ongoing fall in Co2. Greenpeace co-founder and scientist Patrick Moore walked away from Greenpeace in 1986 and has exposed the green movement for fear-mongering and lies. He said that 500 million years ago there was *17 times* more Co2 in the atmosphere than we have today and levels have been falling for hundreds of millions of years. In the last 150 million years Co2 levels in Earth's atmosphere had reduced by *90 percent*. Moore said that by the time humanity began to unlock carbon dioxide from fossil fuels we were at '38 seconds to midnight' and in that sense: 'Humans are [the Earth's] salvation.' Moore made the point that only half the Co2 emitted by fossil fuels stays in the atmosphere and we should remember that all pollution pouring from chimneys that we are told is carbon dioxide is in fact nothing of the kind. It's pollution. Carbon dioxide is an invisible gas.

William Happer, Professor of Physics at Princeton University and long-time government adviser on climate, has emphasised the Co2 deficiency for maximum growth and food production. Greenhouse growers don't add carbon dioxide for a bit of fun. He said that most of the warming in the last 100 years, after the earth emerged from the super-cold period of the 'Little Ice Age' into a natural warming cycle, was over by 1940. Happer said that a peak year for warming in 1988 can be explained by a 'monster El Nino' which is a natural and cyclical warming of the Pacific that has nothing to do with 'climate

change'. He said the effect of Co2 could be compared to painting a wall with red paint in that once two or three coats have been applied it didn't matter how much more you slapped on because the wall will not get much redder. Almost all the effect of the rise in Co2 has already happened, he said, and the volume in the atmosphere would now have to *double* to increase temperature by a single degree. Climate hoaxers know this and they have invented the most ridiculously complicated series of 'feedback' loops to try to overcome this rather devastating fact. You hear puppet Greta going on cluelessly about feedback loops and this is why.

The Sun affects temperature? No you *climate denier*

Some other nonsense to contemplate: Climate graphs show that rises in temperature do not follow rises in Co2 – *it's the other way round* with a lag between the two of some 800 years. If we go back 800 years from present time we hit the Medieval Warm Period when temperatures were higher than now without any industrialisation and this was followed by the Little Ice Age when temperatures plummeted. The world was still emerging from these centuries of serious cold when many climate records began which makes the ever-repeated line of the 'hottest year since records began' meaningless when you are not comparing like with like. The coldest period of the Little Ice Age corresponded with the lowest period of sunspot activity when the Sun was at its least active. Proper scientists will not be at all surprised by this when it confirms the obvious fact that earth temperature is affected by the scale of Sun activity and the energetic power that it subsequently emits; but when is the last time you heard a climate hoaxer talking about the Sun as a source of earth temperature?? Everything has to be focussed on Co2 which makes up just 0.117 percent of so-called greenhouse gases and only a fraction of even that is generated by human activity. The rest is natural. More than *90 percent* of those greenhouse gases are water vapour and clouds (Fig 9). Ban moisture I say. Have you noticed that the climate hoaxers no longer use the polar bear as their promotion image? That's because far from becoming extinct polar

bear communities are stable or thriving. Joe Bastardi, American meteorologist, weather forecaster and outspoken critic of the climate lie, documents in his book *The Climate Chronicles* how weather patterns and events claimed to be evidence of climate change have been happening since long before industrialisation: 'What happened before naturally is happening again, as is to be expected given the cyclical nature of the climate due to the design of the planet.' If you read the detailed background to the climate hoax in my other books you will shake your head and wonder how anyone could believe the crap which has spawned a multi-trillion dollar industry based on absolute garbage (see HIV causes AIDs and Sars-Cov-2 causes 'Covid-19'). Climate and 'Covid' have much in common given they have the same source. They both have the contradictory *everything* factor in which everything is explained by reference to them. It's hot – 'it's climate change'. It's cold – 'it's climate change'. I got a sniffle – 'it's Covid'. I haven't got a sniffle – 'it's Covid'. Not having a sniffle has to be a symptom of 'Covid'. Everything is and not having a sniffle is especially dangerous if you are a slow walker. For sheer audacity I offer you a Cambridge University 'study' that actually linked 'Covid' to 'climate change'. It had to happen eventually. They concluded that climate change played a role in 'Covid-19' spreading from animals to humans because … wait for it … I kid you not … *the two groups were forced closer together as populations grow*. Er, that's it. The whole foundation on which this depended was that 'Bats are the likely zoonotic origin of SARS-CoV-1 and SARS-CoV-2'. Well, they are not. They are nothing to do with it. Apart from bats not being the origin and therefore 'climate change' effects on bats being irrelevant I am in awe of their academic insight. Where would we be without them? Not where we are that's for sure.

Figure 9: The idea that the gas of life is disastrously changing the climate is an insult to brain cell activity.

One other point about the weather is that climate modification is now well advanced and not every major weather event is natural – or earthquake come to that. I cover this subject at some length in other books. China is openly planning a rapid expansion of its weather modification programme which includes changing the climate in an area more than one and a half times the size of India. China used weather manipulation to ensure clear skies during the 2008 Olympics in Beijing. I have quoted from US military documents detailing how to employ weather manipulation as a weapon of war and they did that in the 1960s and 70s during the conflict in Vietnam with Operation Popeye manipulating monsoon rains for military purposes. Why would there be international treaties on weather modification if it wasn't possible? Of course it is. Weather is energetic information and it can be changed.

How was the climate hoax pulled off? See 'Covid'

If you can get billions to believe in a 'virus' that doesn't exist you can get them to believe in human-caused climate change that doesn't exist. Both are being used by the Cult to transform global society in the way it has long planned. Both hoaxes have been achieved in pretty much the same way. First you declare a lie is a fact. There's a

'virus' you call SARS-Cov-2 or humans are warming the planet with their behaviour. Next this becomes, via Cult networks, the foundation of government, academic and science policy and belief. Those who parrot the mantra are given big grants to produce research that confirms the narrative is true and ever more 'symptoms' are added to make the 'virus'/'climate change' sound even more scary. Scientists and researchers who challenge the narrative have their grants withdrawn and their careers destroyed. The media promote the lie as the unquestionable truth and censor those with an alternative view or evidence. A great percentage of the population believe what they are told as the lie becomes an everybody-knows-that and the believing-masses turn on those with a mind of their own. The technique has been used endlessly throughout human history. Wokers are the biggest promotors of the climate lie *and* 'Covid' fascism because their minds are owned by the Cult; their sense of self-righteous self-purity knows no bounds; and they exist in a bubble of reality in which facts are irrelevant and only get in the way of looking without seeing.

Running through all of this like veins in a blue cheese is control of information, which means control of perception, which means control of behaviour, which collectively means control of human society. The Cult owns the global media and Silicon Valley fascists for the simple reason that it *has* to. Without control of information it can't control perception and through that human society. Examine every facet of the Cult agenda and you will see that anything supporting its introduction is never censored while anything pushing back is always censored. I say again: Psychopaths that know why they are doing this must go before Nuremberg trials and those that follow their orders must trot along behind them into the same dock. 'I was just following orders' didn't work the first time and it must not work now. Nuremberg trials must be held all over the world before public juries for politicians, government officials, police, compliant doctors, scientists and virologists, and all Cult operatives such as Gates, Tedros, Fauci, Vallance, Whitty, Ferguson, Zuckerberg, Wojcicki, Brin, Page, Dorsey, the whole damn lot of

them – including, no *especially*, the psychopath psychologists. Without them and the brainless, gutless excuses for journalists that have repeated their lies, none of this could be happening. Nobody can be allowed to escape justice for the psychological and economic Armageddon they are all responsible for visiting upon the human race.

As for the compliant, unquestioning, swathes of humanity, and the self-obsessed, all-knowing ignorance of the Wokers … don't start me. God help their kids. God help their grandkids. God *help them*.

CHAPTER NINE

We must have it? So what is it?

Well I won't back down. No, I won't back down. You can stand me up at the Gates of Hell. But I won't back down
Tom Petty

I will now focus on the genetically-manipulating 'Covid vaccines' which do not meet this official definition of a vaccine by the US Centers for Disease Control (CDC): 'A product that stimulates a person's immune system to produce immunity to a specific disease, protecting the person from that disease.' On that basis 'Covid vaccines' are not a vaccine in that the makers don't even claim they stop infection or transmission.

They are instead part of a multi-levelled conspiracy to change the nature of the human body and what it means to be 'human' and to depopulate an enormous swathe of humanity. What I shall call Human 1.0 is on the cusp of becoming Human 2.0 and for very sinister reasons. Before I get to the 'Covid vaccine' in detail here's some background to vaccines in general. Government regulators do not test vaccines – the makers do – and the makers control which data is revealed and which isn't. Children in America are given 50 vaccine doses by age six and 69 by age 19 and the effect of the whole combined schedule has never been tested. Autoimmune diseases when the immune system attacks its own body have soared in the mass vaccine era and so has disease in general in children and the young. Why wouldn't this be the case when vaccines target the *immune system?* The US government gave Big Pharma drug

companies immunity from prosecution for vaccine death and injury in the 1986 National Childhood Vaccine Injury Act (NCVIA) and since then the government (taxpayer) has been funding compensation for the consequences of Big Pharma vaccines. The criminal and satanic drug giants can't lose and the vaccine schedule has increased dramatically since 1986 for this reason. There is no incentive to make vaccines safe and a big incentive to make money by introducing ever more. Even against a ridiculously high bar to prove vaccine liability, and with the government controlling the hearing in which it is being challenged for compensation, the vaccine court has so far paid out more than $4 billion. These are the vaccines we are told are safe and psychopaths like Zuckerberg censor posts saying otherwise. The immunity law was even justified by a ruling that vaccines by their nature were 'unavoidably unsafe'.

Check out the ingredients of vaccines and you will be shocked if you are new to this. *They put that in children's bodies?? What??* Try aluminium, a brain toxin connected to dementia, aborted foetal tissue and formaldehyde which is used to embalm corpses. World-renowned aluminium expert Christopher Exley had his research into the health effect of aluminium in vaccines shut down by Keele University in the UK when it began taking funding from the Bill and Melinda Gates Foundation. Research when diseases 'eradicated' by vaccines began to decline and you will find the fall began long *before* the vaccine was introduced. Sometimes the fall even plateaued after the vaccine. Diseases like scarlet fever for which there was no vaccine declined in the same way because of environmental and other factors. A perfect case in point is the polio vaccine. Polio began when lead arsenate was first sprayed as an insecticide and residues remained in food products. Spraying started in 1892 and the first US polio epidemic came in Vermont in 1894. The simple answer was to stop spraying, but Rockefeller-created Big Pharma had a better idea. Polio was decreed to be caused by the *poliovirus* which 'spreads from person to person and can infect a person's spinal cord'. Lead arsenate was replaced by the lethal DDT which had the same effect of causing paralysis by damaging the brain and central nervous

system. Polio plummeted when DDT was reduced and then banned, but the vaccine is still given the credit for something it didn't do. Today by far the biggest cause of polio is the vaccines promoted by Bill Gates. Vaccine justice campaigner Robert Kennedy Jr, son of assassinated (by the Cult) US Attorney General Robert Kennedy, wrote:

> In 2017, the World Health Organization (WHO) reluctantly admitted that the global explosion in polio is predominantly vaccine strain. The most frightening epidemics in Congo, Afghanistan, and the Philippines, are all linked to vaccines. In fact, by 2018, 70% of global polio cases were vaccine strain.

Vaccines make fortunes for Cult-owned Gates and Big Pharma while undermining the health and immune systems of the population. We had a glimpse of the mentality behind the Big Pharma cartel with a report on WION (World is One News), an international English language TV station based in India, which exposed the extraordinary behaviour of US drug company Pfizer over its 'Covid vaccine'. The WION report told how Pfizer had made fantastic demands of Argentina, Brazil and other countries in return for its 'vaccine'. These included immunity from prosecution, even for Pfizer negligence, government insurance to protect Pfizer from law suits and handing over as collateral sovereign assets of the country to include Argentina's bank reserves, military bases and embassy buildings. Pfizer demanded the same of Brazil in the form of waiving sovereignty of its assets abroad; exempting Pfizer from Brazilian laws; and giving Pfizer immunity from all civil liability. This is a 'vaccine' developed with government funding. Big Pharma is evil incarnate as a creation of the Cult and all must be handed tickets to Nuremberg.

Phantom 'vaccine' for a phantom 'disease'

I'll expose the 'Covid vaccine' fraud and then go on to the wider background of why the Cult has set out to 'vaccinate' every man, woman and child on the planet for an alleged 'new disease' with a survival rate of 99.77 percent (or more) even by the grotesquely-

manipulated figures of the World Health Organization and Johns Hopkins University. The 'infection' to 'death' ratio is 0.23 to 0.15 percent according to Stanford epidemiologist Dr John Ioannidis and while estimates vary the danger remains tiny. I say that if the truth be told the fake infection to fake death ratio is zero. Never mind all the evidence I have presented here and in *The Answer* that there is no 'virus' let us just focus for a moment on that death-rate figure of say 0.23 percent. The figure includes all those worldwide who have tested positive with a test not testing for the 'virus' and then died within 28 days or even longer of any other cause – *any other cause*. Now subtract all those illusory 'Covid' deaths on the global data sheets from the 0.23 percent. What do you think you would be left with? *Zero*. A vaccination has never been successfully developed for a so-called coronavirus. They have all failed at the animal testing stage when they caused hypersensitivity to what they were claiming to protect against and made the impact of a disease far worse. Cult-owned vaccine corporations got around that problem this time by bypassing animal trials, going straight to humans and making the length of the 'trials' before the public rollout as short as they could get away with. Normally it takes five to ten years or more to develop vaccines that still cause demonstrable harm to many people and that's without including the long-term effects that are never officially connected to the vaccination. 'Covid' non-vaccines have been officially produced and approved in a matter of months from a standing start and part of the reason is that (a) they were developed before the 'Covid' hoax began and (b) they are based on computer programs and not natural sources. Official non-trials were so short that government agencies gave *emergency*, not full, approval. 'Trials' were not even completed and full approval cannot be secured until they are. Public 'Covid vaccination' is actually a *continuation of the trial*. Drug company 'trials' are not scheduled to end until 2023 by which time a lot of people are going to be dead. Data on which government agencies gave this emergency approval was supplied by the Big Pharma corporations themselves in the form of Pfizer/BioNTech, AstraZeneca, Moderna, Johnson & Johnson, and

others, and this is the case with all vaccines. By its very nature *emergency* approval means drug companies do not have to prove that the 'vaccine' is 'safe and effective'. How could they with trials way short of complete? Government regulators only have to *believe* that they *could* be safe and effective. It is criminal manipulation to get products in circulation with no testing worth the name. Agencies giving that approval are infested with Big Pharma-connected place-people and they act in the interests of Big Pharma (the Cult) and not the public about whom they do not give a damn.

More human lab rats

'Covid vaccines' produced in record time by Pfizer/BioNTech and Moderna employ a technique *never approved before for use on humans*. They are known as mRNA 'vaccines' and inject a synthetic version of 'viral' mRNA or 'messenger RNA'. The key is in the term 'messenger'. The body works, or doesn't, on the basis of information messaging. Communications are constantly passing between and within the genetic system and the brain. Change those messages and you change the state of the body and even its very nature and you can change psychology and behaviour by the way the brain processes information. I think you are going to see significant changes in personality and perception of many people who have had the 'Covid vaccine' synthetic potions. Insider Aldous Huxley predicted the following in 1961 and mRNA 'vaccines' can be included in the term 'pharmacological methods':

> There will be, in the next generation or so, a pharmacological method of making people love their servitude, and producing dictatorship without tears, so to speak, producing a kind of painless concentration camp for entire societies, so that people will in fact have their own liberties taken away from them, but rather enjoy it, because they will be distracted from any desire to rebel by propaganda or brainwashing, or brainwashing enhanced by pharmacological methods. And this seems to be the final revolution.

Apologists claim that mRNA synthetic 'vaccines' don't change the DNA genetic blueprint because RNA does not affect DNA only the other way round. This is so disingenuous. A process called 'reverse

transcription' can convert RNA into DNA and be integrated into DNA in the cell nucleus. This was highlighted in December, 2020, by scientists at Harvard and Massachusetts Institute of Technology (MIT). Geneticists report that more than 40 percent of mammalian genomes results from reverse transcription. On the most basic level if messaging changes then that sequence must lead to changes in DNA which is receiving and transmitting those communications. How can introducing synthetic material into cells not change the cells where DNA is located? The process is known as transfection which is defined as 'a technique to insert foreign nucleic acid (DNA or RNA) into a cell, typically with the intention of altering the properties of the cell'. Researchers at the Sloan Kettering Institute in New York found that changes in messenger RNA can deactivate tumour-suppressing proteins and thereby promote cancer. This is what happens when you mess with messaging. 'Covid vaccine' maker Moderna was founded in 2010 by Canadian stem cell biologist Derrick J. Rossi after his breakthrough discovery in the field of transforming and reprogramming stem cells. These are neutral cells that can be programmed to become any cell including sperm cells. Moderna was therefore founded on the principle of genetic manipulation and has never produced any vaccine or drug before its genetically-manipulating synthetic 'Covid' shite. Look at the name – Mode-RNA or Modify-RNA. Another important point is that the US Supreme Court has ruled that genetically-modified DNA, or complementary DNA (cDNA) synthesized in the laboratory from messenger RNA, can be patented and owned. These psychopaths are doing this to the human body.

Cells replicate synthetic mRNA in the 'Covid vaccines' and in theory the body is tricked into making antigens which trigger antibodies to target the 'virus spike proteins' which as Dr Tom Cowan said have *never been seen*. Cut the crap and these 'vaccines' deliver *self-replicating* synthetic material to the cells with the effect of changing human DNA. The more of them you have the more that process is compounded while synthetic material is all the time self-replicating. 'Vaccine'-maker Moderna describes mRNA as 'like

software for the cell' and so they are messing with the body's software. What happens when you change the software in a computer? Everything changes. For this reason the Cult is preparing a production line of mRNA 'Covid vaccines' and a long list of excuses to use them as with all the 'variants' of a 'virus' never shown to exist. The plan is further to transfer the mRNA technique to other vaccines mostly given to children and young people. The cumulative consequences will be a transformation of human DNA through a constant infusion of synthetic genetic material which will kill many and change the rest. Now consider that governments that have given emergency approval for a vaccine that's not a vaccine; never been approved for humans before; had no testing worth the name; and the makers have been given immunity from prosecution for any deaths or adverse effects suffered by the public. The UK government awarded *permanent legal indemnity* to itself and its employees for harm done when a patient is being treated for 'Covid-19' or 'suspected Covid-19'. That is quite a thought when these are possible 'side-effects' from the 'vaccine' (they are not 'side', they are effects) listed by the US Food and Drug Administration:

Guillain-Barre syndrome; acute disseminated encephalomyelitis; transverse myelitis; encephalitis; myelitis; encephalomyelitis; meningoencephalitis; meningitis; encephalopathy; convulsions; seizures; stroke; narcolepsy; cataplexy; anaphylaxis; acute myocardial infarction (heart attack); myocarditis; pericarditis; autoimmune disease; death; implications for pregnancy, and birth outcomes; other acute demyelinating diseases; non anaphylactic allergy reactions; thrombocytopenia ; disseminated intravascular coagulation; venous thromboembolism; arthritis; arthralgia; joint pain; Kawasaki disease; multisystem inflammatory syndrome in children; vaccine enhanced disease. The latter is the way the 'vaccine' has the potential to make diseases far worse than they would otherwise be.

UK doctor and freedom campaigner Vernon Coleman described the conditions in this list as 'all unpleasant, most of them very serious, and you can't get more serious than death'. The thought that anyone at all has had the 'vaccine' in these circumstances is testament to the potential that humanity has for clueless, unquestioning, stupidity and for many that programmed stupidity has already been terminal.

An insider speaks

Dr Michael Yeadon is a former Vice President, head of research and Chief Scientific Adviser at vaccine giant Pfizer. Yeadon worked on the inside of Big Pharma, but that did not stop him becoming a vocal critic of 'Covid vaccines' and their potential for multiple harms, including infertility in women. By the spring of 2021 he went much further and even used the no, no, term 'conspiracy'. When you begin to see what is going on it is impossible not to do so. Yeadon spoke out in an interview with freedom campaigner James Delingpole and I mentioned earlier how he said that no one had samples of 'the virus'. He explained that the mRNA technique originated in the anti-cancer field and ways to turn on and off certain genes which could be advantageous if you wanted to stop cancer growing out of control. 'That's the origin of them. They are a very unusual application, really.' Yeadon said that treating a cancer patient with an aggressive procedure might be understandable if the alternative was dying, but it was quite another thing to use the same technique as a public health measure. Most people involved wouldn't catch the infectious agent you were vaccinating against and if they did they probably wouldn't die:

> If you are really using it as a public health measure you really want to as close as you can get to zero sides-effects … I find it odd that they chose techniques that were really cutting their teeth in the field of oncology and I'm worried that in using gene-based vaccines that have to be injected in the body and spread around the body, get taken up into some cells, and the regulators haven't quite told us which cells they get taken up into … you are going to be generating a wide range of responses … with multiple steps each of which could go well or badly.

I doubt the Cult intends it to go well. Yeadon said that you can put any gene you like into the body through the 'vaccine'. 'You can certainly give them a gene that would do them some harm if you wanted.' I was intrigued when he said that when used in the cancer field the technique could turn genes on and off. I explore this process in *The Answer* and with different genes having different functions you could create mayhem – physically and psychologically – if you turned the wrong ones on and the right ones off. I read reports of an experiment by researchers at the University of Washington's school of computer science and engineering in which they encoded DNA to infect computers. The body is itself a biological computer and if human DNA can inflict damage on a computer why can't the computer via synthetic material mess with the human body? It can. The Washington research team said it was possible to insert malicious malware into 'physical DNA strands' and corrupt the computer system of a gene sequencing machine as it 'reads gene letters and stores them as binary digits 0 and 1'. They concluded that hackers could one day use blood or spit samples to access computer systems and obtain sensitive data from police forensics labs or infect genome files. It is at this level of digital interaction that synthetic 'vaccines' need to be seen to get the full picture and that will become very clear later on. Michael Yeadon said it made no sense to give the 'vaccine' to younger people who were in no danger from the 'virus'. What was the benefit? It was all downside with potential effects:

> The fact that my government in what I thought was a civilised, rational country, is raining [the 'vaccine'] on people in their 30s and 40s, even my children in their 20s, they're getting letters and phone calls, I know this is not right and any of you doctors who are vaccinating you know it's not right, too. They are not at risk. They are not at risk from the disease, so you are now hoping that the side-effects are so rare that you get away with it. You don't give new technology … that you don't understand to 100 percent of the population.

Blood clot problems with the AstraZeneca 'vaccine' have been affecting younger people to emphasise the downside risks with no benefit. AstraZeneca's version, produced with Oxford University, does not use mRNA, but still gets its toxic cocktail inside cells where

it targets DNA. The Johnson & Johnson 'vaccine' which uses a similar technique has also produced blood clot effects to such an extent that the United States paused its use at one point. They are all 'gene therapy' (cell modification) procedures and not 'vaccines'. The truth is that once the content of these injections enter cells we have no idea what the effect will be. People can speculate and some can give very educated opinions and that's good. In the end, though, only the makers know what their potions are designed to do and even they won't know every last consequence. Michael Yeadon was scathing about doctors doing what they knew to be wrong. 'Everyone's mute', he said. Doctors in the NHS must know this was not right, coming into work and injecting people. 'I don't know how they sleep at night. I know I couldn't do it. I know that if I were in that position I'd have to quit.' He said he knew enough about toxicology to know this was not a good risk-benefit. Yeadon had spoken to seven or eight university professors and all except two would not speak out publicly. Their universities had a policy that no one said anything that countered the government and its medical advisors. They were afraid of losing their government grants. This is how intimidation has been used to silence the truth at every level of the system. I say silence, but these people could still speak out if they made that choice. Yeadon called them 'moral cowards' – 'This is about your children and grandchildren's lives and you have just buggered off and left it.'

'Variant' nonsense

Some of his most powerful comments related to the alleged 'variants' being used to instil more fear, justify more lockdowns, and introduce more 'vaccines'. He said government claims about 'variants' were nonsense. He had checked the alleged variant 'codes' and they were 99.7 percent identical to the 'original'. This was the human identity difference equivalent to putting a baseball cap on and off or wearing it the other way round. A 0.3 percent difference would make it impossible for that 'variant' to escape immunity from the 'original'. This made no sense of having new 'vaccines' for

'variants'. He said there would have to be at least a *30 percent* difference for that to be justified and even then he believed the immune system would still recognise what it was. Gates-funded 'variant modeller' and 'vaccine'-pusher John Edmunds might care to comment. Yeadon said drug companies were making new versions of the 'vaccine' as a 'top up' for 'variants'. Worse than that, he said, the 'regulators' around the world like the MHRA in the UK had got together and agreed that because 'vaccines' for 'variants' were so similar to the first 'vaccines' *they did not have to do safety studies*. How transparently sinister that is. This is when Yeadon said: 'There is a conspiracy here.' There was no need for another vaccine for 'variants' and yet we were told that there was and the country had shut its borders because of them. 'They are going into hundreds of millions of arms without passing 'go' or any regulator. Why did they do that? Why did they pick this method of making the vaccine?'

The reason had to be something bigger than that it seemed and 'it's not protection against the virus'. It's was a far bigger project that meant politicians and advisers were willing to do things and not do things that knowingly resulted in avoidable deaths – 'that's already happened when you think about lockdown and deprivation of health care for a year.' He spoke of people prepared to do something that results in the avoidable death of their fellow human beings and it not bother them. This is the penny-drop I have been working to get across for more than 30 years – the level of pure evil we are dealing with. Yeadon said his friends and associates could not believe there could be that much evil, but he reminded them of Stalin, Pol Pot and Hitler and of what Stalin had said: 'One death is a tragedy. A million? A statistic.' He could not think of a benign explanation for why you need top-up vaccines 'which I'm sure you don't' and for the regulators 'to just get out of the way and wave them through'. Why would the regulators do that when they were still wrestling with the dangers of the 'parent' vaccine? He was clearly shocked by what he had seen since the 'Covid' hoax began and now he was thinking the previously unthinkable:

> If you wanted to depopulate a significant proportion of the world and to do it in a way that doesn't involve destruction of the environment with nuclear weapons, poisoning everyone with anthrax or something like that, and you wanted plausible deniability while you had a multi-year infectious disease crisis, I actually don't think you could come up with a better plan of work than seems to be in front of me. I can't say that's what they are going to do, but I can't think of a benign explanation why they are doing it.

He said he never thought that they would get rid of 99 percent of humans, but now he wondered. 'If you wanted to that this would be a hell of a way to do it – it would be unstoppable folks.' Yeadon had concluded that those who submitted to the 'vaccine' would be allowed to have some kind of normal life (but for how long?) while screws were tightened to coerce and mandate the last few percent. 'I think they'll put the rest of them in a prison camp. I wish I was wrong, but I don't think I am.' Other points he made included: There were no coronavirus vaccines then suddenly they all come along at the same time; we have no idea of the long term affect with trials so short; coercing or forcing people to have medical procedures is against the Nuremberg Code instigated when the Nazis did just that; people should at least delay having the 'vaccine'; a quick Internet search confirms that masks don't reduce respiratory viral transmission and 'the government knows that'; they have smashed civil society and they know that, too; two dozen peer-reviewed studies show no connection between lockdown and reducing deaths; he knew from personal friends the elite were still flying around and going on holiday while the public were locked down; the elite were not having the 'vaccines'. He was also asked if 'vaccines' could be made to target difference races. He said he didn't know, but the document by the Project for the New American Century in September, 2000, said developing 'advanced forms of biological warfare that can target *specific genotypes* may transform biological warfare from the realm of terror to a politically useful tool.' Oh, they're evil all right. Of that we can be *absolutely* sure.

Another cull of old people

We have seen from the CDC definition that the mRNA 'Covid vaccine' is not a vaccine and nor are the others that *claim* to reduce 'severity of symptoms' in *some* people, but not protect from infection or transmission. What about all the lies about returning to 'normal' if people were 'vaccinated'? If they are not claimed to stop infection and transmission of the alleged 'virus', how does anything change? This was all lies to manipulate people to take the jabs and we are seeing that now with masks and distancing still required for the 'vaccinated'. How did they think that elderly people with fragile health and immune responses were going to be affected by infusing their cells with synthetic material and other toxic substances? They *knew* that in the short and long term it would be devastating and fatal as the culling of the old that began with the first lockdowns was continued with the 'vaccine'. Death rates in care homes soared immediately residents began to be 'vaccinated' – infused with synthetic material. Brave and committed whistleblower nurses put their careers at risk by exposing this truth while the rest kept their heads down and their mouths shut to put their careers before those they are supposed to care for. A long-time American Certified Nursing Assistant who gave his name as James posted a video in which he described emotionally what happened in his care home when vaccination began. He said that during 2020 very few residents were sick with 'Covid' and no one died during the entire year; but shortly after the Pfizer mRNA injections 14 people died within two weeks and many others were near death. 'They're dropping like flies', he said. Residents who walked on their own before the shot could no longer and they had lost their ability to conduct an intelligent conversation. The home's management said the sudden deaths were caused by a 'super-spreader' of 'Covid-19'. Then how come, James asked, that residents who refused to take the injections were not sick? It was a case of inject the elderly with mRNA synthetic potions and blame their illness and death that followed on the 'virus'. James described what was happening in care homes as 'the greatest crime of genocide this country has ever seen'. Remember the NHS staff nurse from earlier who used the same

word ‘genocide’ for what was happening with the ‘vaccines’ and that it was an ‘act of human annihilation’. A UK care home whistleblower told a similar story to James about the effect of the ‘vaccine’ in deaths and ‘outbreaks’ of illness dubbed ‘Covid’ after getting the jab. She told how her care home management and staff had zealously imposed government regulations and no one was allowed to even question the official narrative let alone speak out against it. She said the NHS was even worse. Again we see the results of reframing. A worker at a local care home where I live said they had not had a single case of ‘Covid’ there for almost a year and when the residents were ‘vaccinated’ they had 19 positive cases in two weeks with eight dying.

It’s not the ‘vaccine’ – honest

The obvious cause and effect was being ignored by the media and most of the public. Australia’s health minister Greg Hunt (a former head of strategy at the World Economic Forum) was admitted to hospital after he had the ‘vaccine’. He was suffering according to reports from the skin infection ‘cellulitis’ and it must have been a severe case to have warranted days in hospital. Immediately the authorities said this was nothing to do with the ‘vaccine’ when an effect of some vaccines is a ‘cellulitis-like reaction’. We had families of perfectly healthy old people who died after the ‘vaccine’ saying that if only they had been given the ‘vaccine’ earlier they would still be alive. As a numbskull rating that is off the chart. A father of four ‘died of Covid’ at aged 48 when he was taken ill two days after having the ‘vaccine’. The man, a health administrator, had been ‘shielding during the pandemic’ and had ‘not really left the house’ until he went for the ‘vaccine’. Having the ‘vaccine’ and then falling ill and dying does not seem to have qualified as a possible cause and effect and ‘Covid-19’ went on his death certificate. His family said they had no idea how he ‘caught the virus’. A family member said: ‘Tragically, it could be that going for a vaccination ultimately led to him catching Covid …The sad truth is that they are never going to know where it came from.’ The family warned people to remember

that the virus still existed and was 'very real'. So was their stupidity. Nurses and doctors who had the first round of the 'vaccine' were collapsing, dying and ending up in a hospital bed while they or their grieving relatives were saying they'd still have the 'vaccine' again despite what happened. I kid you not. You mean if your husband returned from the dead he'd have the same 'vaccine' again that killed him??

Doctors at the VCU Medical Center in Richmond, Virginia, said the Johnson & Johnson 'vaccine' was to blame for a man's skin peeling off. Patient Richard Terrell said: 'It all just happened so fast. My skin peeled off. It's still coming off on my hands now.' He said it was stinging, burning and itching and when he bent his arms and legs it was very painful with 'the skin swollen and rubbing against itself'. Pfizer/BioNTech and Moderna vaccines use mRNA to change the cell while the Johnson & Johnson version uses DNA in a process similar to AstraZeneca's technique. Johnson & Johnson and AstraZeneca have both had their 'vaccines' paused by many countries after causing serious blood problems. Terrell's doctor Fnu Nutan said he could have died if he hadn't got medical attention. It sounds terrible so what did Nutan and Terrell say about the 'vaccine' now? Oh, they still recommend that people have it. A nurse in a hospital bed 40 minutes after the vaccination and unable to swallow due to throat swelling was told by a doctor that he lost mobility in his arm for 36 hours following the vaccination. What did he say to the ailing nurse? 'Good for you for getting the vaccination.' We are dealing with a serious form of cognitive dissonance madness in both public and medical staff. There is a remarkable correlation between those having the 'vaccine' and trumpeting the fact and suffering bad happenings shortly afterwards. Witold Rogiewicz, a Polish doctor, made a video of his 'vaccination' and ridiculed those who were questioning its safety and the intentions of Bill Gates: 'Vaccinate yourself to protect yourself, your loved ones, friends and also patients. And to mention quickly I have info for anti-vaxxers and anti-Coviders if you want to contact Bill Gates you can do this through me.' He further ridiculed the dangers of 5G. Days later he

was dead, but naturally the vaccination wasn't mentioned in the verdict of 'heart attack'.

Lies, lies and more lies

So many members of the human race have slipped into extreme states of insanity and unfortunately they include reframed doctors and nursing staff. Having a 'vaccine' and dying within minutes or hours is not considered a valid connection while death from any cause within 28 days or longer of a positive test with a test not testing for the 'virus' means 'Covid-19' goes on the death certificate. How could that 'vaccine'-death connection not have been made except by calculated deceit? US figures in the initial rollout period to February 12th, 2020, revealed that a third of the deaths reported to the CDC after 'Covid vaccines' happened within 48 hours. Five men in the UK suffered an 'extremely rare' blood clot problem after having the AstraZeneca 'vaccine', but no causal link was established said the Gates-funded Medicines and Healthcare products Regulatory Agency (MHRA) which had given the 'vaccine' emergency approval to be used. Former Pfizer executive Dr Michael Yeadon explained in his interview how the procedures could cause blood coagulation and clots. People who should have been at no risk were dying from blood clots in the brain and he said he had heard from medical doctor friends that people were suffering from skin bleeding and massive headaches. The AstraZeneca 'shot' was stopped by some 20 countries over the blood clotting issue and still the corrupt MHRA, the European Medicines Agency (EMA) and the World Health Organization said that it should continue to be given even though the EMA admitted that it 'still cannot rule out definitively' a link between blood clotting and the 'vaccine'. Later Marco Cavaleri, head of EMA vaccine strategy, said there was indeed a clear link between the 'vaccine' and thrombosis, but they didn't know why. So much for the trials showing the 'vaccine' is safe. Blood clots were affecting younger people who would be under virtually no danger from 'Covid' even if it existed which makes it all the more stupid and sinister.

The British government responded to public alarm by wheeling out June Raine, the terrifyingly weak infant school headmistress sound-alike who heads the UK MHRA drug 'regulator'. The idea that she would stand up to Big Pharma and government pressure is laughable and she told us that all was well in the same way that she did when allowing untested, never-used-on-humans-before, genetically-manipulating 'vaccines' to be exposed to the public in the first place. Mass lying is the new normal of the 'Covid' era. The MHRA later said 30 cases of rare blood clots had by then been connected with the AstraZeneca 'vaccine' (that means a lot more in reality) while stressing that the benefits of the jab in preventing 'Covid-19' outweighed any risks. A more ridiculous and disingenuous statement with callous disregard for human health it is hard to contemplate. Immediately after the mendacious 'all-clears' two hospital workers in Denmark experienced blood clots and cerebral haemorrhaging following the AstraZeneca jab and one died. Top Norwegian health official Pål Andre Holme said the 'vaccine' was the only common factor: 'There is nothing in the patient history of these individuals that can give such a powerful immune response … I am confident that the antibodies that we have found are the cause, and I see no other explanation than it being the vaccine which triggers it.' Strokes, a clot or bleed in the brain, were clearly associated with the 'vaccine' from word of mouth and whistleblower reports. Similar consequences followed with all these 'vaccines' that we were told were so safe and as the numbers grew by the day it was clear we were witnessing human carnage.

Learning the hard way

A woman interviewed by UKColumn told how her husband suffered dramatic health effects after the vaccine when he'd been in good health all his life. He went from being a little unwell to losing all feeling in his legs and experiencing 'excruciating pain'. Misdiagnosis followed twice at Accident and Emergency (an 'allergy' and 'sciatica') before he was admitted to a neurology ward where doctors said his serious condition had been caused by the

‘vaccine’. Another seven ‘vaccinated’ people were apparently being treated on the same ward for similar symptoms. The woman said he had the ‘vaccine’ because they believed media claims that it was safe. ‘I didn’t think the government would give out a vaccine that does this to somebody; I believed they would be bringing out a vaccination that would be safe.’ What a tragic way to learn that lesson. Another woman posted that her husband was transporting stroke patients to hospital on almost every shift and when he asked them if they had been ‘vaccinated’ for ‘Covid’ they all replied ‘yes’. One had a ‘massive brain bleed’ the day after his second dose. She said her husband reported the ‘just been vaccinated’ information every time to doctors in A and E only for them to ignore it, make no notes and appear annoyed that it was even mentioned. This particular report cannot be verified, but it expresses a common theme that confirms the monumental underreporting of ‘vaccine’ consequences. Interestingly as the ‘vaccines’ and their brain blood clot/stroke consequences began to emerge the UK National Health Service began a publicity campaign telling the public what to do in the event of a stroke. A Scottish NHS staff nurse who quit in disgust in March, 2021, said:

> I have seen traumatic injuries from the vaccine, they’re not getting reported to the yellow card [adverse reaction] scheme, they’re treating the symptoms, not asking why, why it’s happening. It’s just treating the symptoms and when you speak about it you’re dismissed like you’re crazy, I’m not crazy, I’m not crazy because every other colleague I’ve spoken to is terrified to speak out, they’ve had enough.

Videos appeared on the Internet of people uncontrollably shaking after the ‘vaccine’ with no control over muscles, limbs and even their face. A Scottish mother broke out in a severe rash all over her body almost immediately after she was given the AstraZeneca ‘vaccine’. The pictures were horrific. Leigh King, a 41-year-old hairdresser from Lanarkshire said: ‘Never in my life was I prepared for what I was about to experience … My skin was so sore and constantly hot … I have never felt pain like this …’ But don’t you worry, the ‘vaccine’ is perfectly safe. Then there has been the effect on medical

staff who have been pressured to have the 'vaccine' by psychopathic 'health' authorities and government. A London hospital consultant who gave the name K. Polyakova wrote this to the *British Medical Journal* or *BMJ*:

> I am currently struggling with … the failure to report the reality of the morbidity caused by our current vaccination program within the health service and staff population. The levels of sickness after vaccination is unprecedented and staff are getting very sick and some with neurological symptoms which is having a huge impact on the health service function. Even the young and healthy are off for days, some for weeks, and some requiring medical treatment. Whole teams are being taken out as they went to get vaccinated together.
>
> Mandatory vaccination in this instance is stupid, unethical and irresponsible when it comes to protecting our staff and public health. We are in the voluntary phase of vaccination, and encouraging staff to take an unlicensed product that is impacting on their immediate health … it is clearly stated that these vaccine products do not offer immunity or stop transmission. In which case why are we doing it?

Not to protect health that's for sure. Medical workers are lauded by governments for agenda reasons when they couldn't give a toss about them any more than they can for the population in general. Schools across America faced the same situation as they closed due to the high number of teachers and other staff with bad reactions to the Pfizer/BioNTech, Moderna, and Johnson & Johnson 'Covid vaccines' all of which were linked to death and serious adverse effects. The *BMJ* took down the consultant's comments pretty quickly on the grounds that they were being used to spread 'disinformation'. They were exposing the truth about the 'vaccine' was the real reason. The cover-up is breathtaking.

Hiding the evidence

The scale of the 'vaccine' death cover-up worldwide can be confirmed by comparing official figures with the personal experience of the public. I heard of many people in my community who died immediately or soon after the vaccine that would never appear in the media or even likely on the official totals of 'vaccine' fatalities and adverse reactions when only about ten percent are estimated to be

reported and I have seen some estimates as low as one percent in a Harvard study. In the UK alone by April 29th, 2021, some 757,654 adverse reactions had been officially reported from the Pfizer/BioNTech, Oxford/AstraZeneca and Moderna 'vaccines' with more than a thousand deaths linked to jabs and that means an estimated ten times this number in reality from a ten percent reporting rate percentage. That's seven million adverse reactions and 10,000 potential deaths and a one percent reporting rate would be ten times *those* figures. In 1976 the US government pulled the swine flu vaccine after *53* deaths. The UK data included a combined 10,000 eye disorders from the 'Covid vaccines' with more than 750 suffering visual impairment or blindness and again multiply by the estimated reporting percentages. As 'Covid cases' officially fell hospitals virtually empty during the 'Covid crisis' began to fill up with a range of other problems in the wake of the 'vaccine' rollout. The numbers across America have also been catastrophic. Deaths linked to *all* types of vaccine increased by *6,000 percent* in the first quarter of 2021 compared with 2020. A 39-year-old woman from Ogden, Utah, died four days after receiving a second dose of Moderna's 'Covid vaccine' when her liver, heart and kidneys all failed despite the fact that she had no known medical issues or conditions. Her family sought an autopsy, but Dr Erik Christensen, Utah's chief medical examiner, said proving vaccine injury as a cause of death almost never happened. He could think of only one instance where an autopsy would name a vaccine as the official cause of death and that would be anaphylaxis where someone received a vaccine and died almost instantaneously. 'Short of that, it would be difficult for us to definitively say this is the vaccine,' Christensen said. If that is true this must be added to the estimated ten percent (or far less) reporting rate of vaccine deaths and serious reactions and the conclusion can only be that vaccine deaths and serious reactions – including these 'Covid' potions' – are phenomenally understated in official figures. The same story can be found everywhere. Endless accounts of deaths and serious reactions among the public, medical

and care home staff while official figures did not even begin to reflect this.

Professional script-reader Dr David Williams, a 'top public-health official' in Ontario, Canada, insulted our intelligence by claiming only four serious adverse reactions and no deaths from the more than 380,000 vaccine doses then given. This bore no resemblance to what people knew had happened in their owns circles and we had Dirk Huyer in charge of getting millions vaccinated in Ontario while at the same time he was Chief Coroner for the province investigating causes of death including possible death from the vaccine. An aide said he had stepped back from investigating deaths, but evidence indicated otherwise. Rosemary Frei, who secured a Master of Science degree in molecular biology at the Faculty of Medicine at Canada's University of Calgary before turning to investigative journalism, was one who could see that official figures for 'vaccine' deaths and reactions made no sense. She said that doctors seldom reported adverse events and when people got really sick or died after getting a vaccination they would attribute that to anything except the vaccines. It had been that way for years and anyone who wondered aloud whether the 'Covid vaccines' or other shots cause harm is immediately branded as 'anti-vax' and 'anti-science'. This was 'career-threatening' for health professionals. Then there was the huge pressure to support the push to 'vaccinate' billions in the quickest time possible. Frei said:

> So that's where we're at today. More than half a million vaccine doses have been given to people in Ontario alone. The rush is on to vaccinate all 15 million of us in the province by September. And the mainstream media are screaming for this to be sped up even more. That all adds up to only a very slim likelihood that we're going to be told the truth by officials about how many people are getting sick or dying from the vaccines.

What is true of Ontario is true of everywhere.

They KNEW – and still did it

The authorities knew what was going to happen with multiple deaths and adverse reactions. The UK government's Gates-funded

and Big Pharma-dominated Medicines and Healthcare products Regulatory Agency (MHRA) hired a company to employ AI in compiling the projected reactions to the 'vaccine' that would otherwise be uncountable. The request for applications said: 'The MHRA urgently seeks an Artificial Intelligence (AI) software tool to process the expected high volume of Covid-19 vaccine Adverse Drug Reaction …' This was from the agency, headed by the disingenuous June Raine, that gave the 'vaccines' emergency approval and the company was hired before the first shot was given. 'We are going to kill and maim you – is that okay?' 'Oh, yes, perfectly fine – I'm very grateful, thank you, doctor.' The range of 'Covid vaccine' adverse reactions goes on for page after page in the MHRA criminally underreported 'Yellow Card' system and includes affects to eyes, ears, skin, digestion, blood and so on. Raine's MHRA amazingly claimed that the 'overall safety experience … is so far as expected from the clinical trials'. The death, serious adverse effects, deafness and blindness were *expected?* When did they ever mention that? If these human tragedies were expected then those that gave approval for the use of these 'vaccines' must be guilty of crimes against humanity including murder – a definition of which is 'killing a person with malice aforethought or with recklessness manifesting extreme indifference to the value of human life.' People involved at the MHRA, the CDC in America and their equivalent around the world must go before Nuremberg trials to answer for their callous inhumanity. We are only talking here about the immediate effects of the 'vaccine'. The longer-term impact of the DNA synthetic manipulation is the main reason they are so hysterically desperate to inoculate the entire global population in the shortest possible time.

Africa and the developing world are a major focus for the 'vaccine' depopulation agenda and a mass vaccination sales-pitch is underway thanks to caring people like the Rockefellers and other Cult assets. The Rockefeller Foundation, which pre-empted the 'Covid pandemic' in a document published in 2010 that 'predicted' what happened a decade later, announced an initial $34.95 million grant in February, 2021, 'to ensure more equitable access to Covid-19

testing and vaccines' among other things in Africa in collaboration with '24 organizations, businesses, and government agencies'. The pan-Africa initiative would focus on 10 countries: Burkina Faso, Ethiopia, Ghana, Kenya, Nigeria, Rwanda, South Africa, Tanzania, Uganda, and Zambia'. Rajiv Shah, President of the Rockefeller Foundation and former administrator of CIA-controlled USAID, said that if Africa was not mass-vaccinated (to change the DNA of its people) it was a 'threat to all of humanity' and not fair on Africans. When someone from the Rockefeller Foundation says they want to do something to help poor and deprived people and countries it is time for a belly-laugh. They are doing this out of the goodness of their 'heart' because 'vaccinating' the entire global population is what the 'Covid' hoax set out to achieve. Official 'decolonisation' of Africa by the Cult was merely a prelude to financial colonisation on the road to a return to physical colonisation. The 'vaccine' is vital to that and the sudden and convenient death of the 'Covid' sceptic president of Tanzania can be seen in its true light. A lot of people in Africa are aware that this is another form of colonisation and exploitation and they need to stand their ground.

The 'vaccine is working' scam

A potential problem for the Cult was that the 'vaccine' is meant to change human DNA and body messaging and not to protect anyone from a 'virus' never shown to exist. The vaccine couldn't work because it was not designed to work and how could they make it *appear* to be working so that more people would have it? This was overcome by lowering the amplification rate of the PCR test to produce fewer 'cases' and therefore fewer 'deaths'. Some of us had been pointing out since March, 2020, that the amplification rate of the test not testing for the 'virus' had been made artificially high to generate positive tests which they could call 'cases' to justify lockdowns. The World Health Organization recommended an absurdly high 45 amplification cycles to ensure the high positives required by the Cult and then remained silent on the issue until January 20th, 2021 – Biden's Inauguration Day. This was when the

'vaccinations' were seriously underway and on that day the WHO recommended after discussions with America's CDC that laboratories *lowered their testing amplification*. Dr David Samadi, a certified urologist and health writer, said the WHO was encouraging all labs to reduce their cycle count for PCR tests. He said the current cycle was much too high and was 'resulting in any particle being declared a positive case'. Even one mainstream news report I saw said this meant the number of 'Covid' infections may have been 'dramatically inflated'. Oh, just a little bit. The CDC in America issued new guidance to laboratories in April, 2021, to use 28 cycles *but only for 'vaccinated' people*. The timing of the CDC/WHO interventions were cynically designed to make it appear the 'vaccines' were responsible for falling cases and deaths when the real reason can be seen in the following examples. New York's state lab, the Wadsworth Center, identified 872 positive tests in July, 2020, based on a threshold of 40 cycles. When the figure was lowered to 35 cycles *43 percent* of the 872 were no longer 'positives'. At 30 cycles the figure was 63 percent. A Massachusetts lab found that between *85 to 90 percent* of people who tested positive in July with a cycle threshold of 40 would be negative at 30 cycles, Ashish Jha, MD, director of the Harvard Global Health Institute, said: 'I'm really shocked that it could be that high … Boy, does it really change the way we need to be thinking about testing.' I'm shocked that I could see the obvious in the spring of 2020, with no medical background, and most medical professionals still haven't worked it out. No, that's not shocking – it's terrifying.

Three weeks after the WHO directive to lower PCR cycles the London *Daily Mail* ran this headline: 'Why ARE Covid cases plummeting? New infections have fallen 45% in the US and 30% globally in the past 3 weeks but experts say vaccine is NOT the main driver because only 8% of Americans and 13% of people worldwide have received their first dose.' They acknowledged that the drop could not be attributed to the 'vaccine', but soon this morphed throughout the media into the 'vaccine' has caused cases and deaths to fall when it was the PCR threshold. In December, 2020, there was

chaos at English Channel ports with truck drivers needing negative 'Covid' tests before they could board a ferry home for Christmas. The government wanted to remove the backlog as fast as possible and they brought in troops to do the 'testing'. Out of 1,600 drivers just *36* tested positive and the rest were given the all clear to cross the Channel. I guess the authorities thought that 36 was the least they could get away with without the unquestioning catching on. The amplification trick which most people believed in the absence of information in the mainstream applied more pressure on those refusing the 'vaccine' to succumb when it 'obviously worked'. The truth was the exact opposite with deaths in care homes soaring with the 'vaccine' and in Israel the term used was 'skyrocket'. A re-analysis of published data from the Israeli Health Ministry led by Dr Hervé Seligmann at the Medicine Emerging Infectious and Tropical Diseases at Aix-Marseille University found that Pfizer's 'Covid vaccine' killed 'about 40 times more [elderly] people than the disease itself would have killed' during a five-week vaccination period and *260 times* more younger people than would have died from the 'virus' even according to the manipulated 'virus' figures. Dr Seligmann and his co-study author, Haim Yativ, declared after reviewing the Israeli 'vaccine' death data: 'This is a new Holocaust.'

Then, in mid-April, 2021, after vast numbers of people worldwide had been 'vaccinated', the story changed with clear coordination. The UK government began to prepare the ground for more future lockdowns when Nuremberg-destined Boris Johnson told yet another whopper. He said that cases had fallen because of *lockdowns* not 'vaccines'. Lockdowns are irrelevant when *there is no 'virus'* and the test and fraudulent death certificates are deciding the number of 'cases' and 'deaths'. Study after study has shown that lockdowns don't work and instead kill and psychologically destroy people. Meanwhile in the United States Anthony Fauci and Rochelle Walensky, the ultra-Zionist head of the CDC, peddled the same line. More lockdown was the answer and not the 'vaccine', a line repeated on cue by the moron that is Canadian Prime Minister Justin Trudeau. Why all the hysteria to get everyone 'vaccinated' if lockdowns and

not 'vaccines' made the difference? None of it makes sense on the face of it. Oh, but it does. The Cult wants lockdowns *and* the 'vaccine' and if the 'vaccine' is allowed to be seen as the total answer lockdowns would no longer be justified when there are still livelihoods to destroy. 'Variants' and renewed upward manipulation of PCR amplification are planned to instigate never-ending lockdown *and* more 'vaccines'.

You *must* have it – we're desperate

Israel, where the Jewish and Arab population are ruled by the Sabbatian Cult, was the front-runner in imposing the DNA-manipulating 'vaccine' on its people to such an extent that Jewish refusers began to liken what was happening to the early years of Nazi Germany. This would seem to be a fantastic claim. Why would a government of Jewish people be acting like the Nazis did? If you realise that the Sabbatian Cult was behind the Nazis and that Sabbatians hate Jews the pieces start to fit and the question of why a 'Jewish' government would treat Jews with such callous disregard for their lives and freedom finds an answer. Those controlling the government of Israel *aren't Jewish* – they're Sabbatian. Israeli lawyer Tamir Turgal was one who made the Nazi comparison in comments to German lawyer Reiner Fuellmich who is leading a class action lawsuit against the psychopaths for crimes against humanity. Turgal described how the Israeli government was vaccinating children and pregnant women on the basis that there was no evidence that this was dangerous when they had no evidence that it *wasn't* dangerous either. They just had no evidence. This was medical experimentation and Turgal said this breached the Nuremberg Code about medical experimentation and procedures requiring informed consent and choice. Think about that. A Nuremberg Code developed because of Nazi experimentation on Jews and others in concentration camps by people like the evil-beyond-belief Josef Mengele is being breached by the *Israeli* government; but when you know that it's a *Sabbatian* government along with its intelligence and military agencies like Mossad, Shin Bet and the Israeli Defense Forces, and that Sabbatians

were the force behind the Nazis, the kaleidoscope comes into focus. What have we come to when Israeli Jews are suing their government for violating the Nuremberg Code by essentially making Israelis subject to a medical experiment using the controversial 'vaccines'? It's a shocker that this has to be done in the light of what happened in Nazi Germany. The Anshe Ha-Emet, or 'People of the Truth', made up of Israeli doctors, lawyers, campaigners and public, have launched a lawsuit with the International Criminal Court. It says:

> When the heads of the Ministry of Health as well as the prime minister presented the vaccine in Israel and began the vaccination of Israeli residents, the vaccinated were not advised, that, in practice, they are taking part in a medical experiment and that their consent is required for this under the Nuremberg Code.

The irony is unbelievable, but easily explained in one word: Sabbatians. The foundation of Israeli 'Covid' apartheid is the 'green pass' or 'green passport' which allows Jews and Arabs who have had the DNA-manipulating 'vaccine' to go about their lives – to work, fly, travel in general, go to shopping malls, bars, restaurants, hotels, concerts, gyms, swimming pools, theatres and sports venues, while non-'vaccinated' are banned from all those places and activities. Israelis have likened the 'green pass' to the yellow stars that Jews in Nazi Germany were forced to wear – the same as the yellow stickers that a branch of UK supermarket chain Morrisons told exempt mask-wears they had to display when shopping. How very sensitive. The Israeli system is blatant South African-style apartheid on the basis of compliance or non-compliance to fascism rather than colour of the skin. How appropriate that the Sabbatian Israeli government was so close to the pre-Mandela apartheid regime in Pretoria. The Sabbatian-instigated 'vaccine passport' in Israel is planned for everywhere. Sabbatians struck a deal with Pfizer that allowed them to lead the way in the percentage of a national population infused with synthetic material and the result was catastrophic. Israeli freedom activist Shai Dannon told me how chairs were appearing on beaches that said 'vaccinated only'. Health Minister Yuli Edelstein said that anyone unwilling or unable to get

the jabs that 'confer immunity' will be 'left behind'. The man's a liar. Not even the makers claim the 'vaccines' confer immunity. When you see those figures of 'vaccine' deaths these psychopaths were saying that you must take the chance the 'vaccine' will kill you or maim you while knowing it will change your DNA or lockdown for you will be permanent. That's fascism. The Israeli parliament passed a law to allow personal information of the non-vaccinated to be shared with local and national authorities for three months. This was claimed by its supporters to be a way to 'encourage' people to be vaccinated. Hadas Ziv from Physicians for Human Rights described this as a 'draconian law which crushed medical ethics and the patient rights'. But that's the idea, the Sabbatians would reply.

Your papers, please

Sabbatian Israel was leading what has been planned all along to be a global 'vaccine pass' called a 'green passport' without which you would remain in permanent lockdown restriction and unable to do anything. This is how badly – *desperately* – the Cult is to get everyone 'vaccinated'. The term and colour 'green' was not by chance and related to the psychology of fusing the perception of the green climate hoax with the 'Covid' hoax and how the 'solution' to both is the same Great Reset. Lying politicians, health officials and psychologists denied there were any plans for mandatory vaccinations or restrictions based on vaccinations, but they knew that was exactly what was meant to happen with governments of all countries reaching agreements to enforce a global system. 'Free' Denmark and 'free' Sweden unveiled digital vaccine certification. Cyprus, Czech Republic, Estonia, Greece, Hungary, Iceland, Italy, Poland, Portugal, Slovakia, and Spain have all committed to a vaccine passport system and the rest including the whole of the EU would follow. The satanic UK government will certainly go this way despite mendacious denials and at the time of writing it is trying to manipulate the public into having the 'vaccine' so they could go abroad on a summer holiday. How would that work without something to prove you had the synthetic toxicity injected into you?

Documents show that the EU's European Commission was moving towards 'vaccine certificates' in 2018 and 2019 before the 'Covid' hoax began. They knew what was coming. Abracadabra – Ursula von der Leyen, the German President of the Commission, announced in March, 2021, an EU 'Digital Green Certificate' – green again – to track the public's 'Covid status'. The passport sting is worldwide and the Far East followed the same pattern with South Korea ruling that only those with 'vaccination' passports – again the *green* pass – would be able to 'return to their daily lives'.

Bill Gates has been preparing for this 'passport' with other Cult operatives for years and beyond the paper version is a Gates-funded 'digital tattoo' to identify who has been vaccinated and who hasn't. The 'tattoo' is reported to include a substance which is externally readable to confirm who has been vaccinated. This is a bio-luminous light-generating enzyme (think fireflies) called … *Luciferase*. Yes, named after the Cult 'god' Lucifer the 'light bringer' of whom more to come. Gates said he funded the readable tattoo to ensure children in the developing world were vaccinated and no one was missed out. He cares so much about poor kids as we know. This was just the cover story to develop a vaccine tagging system for everyone on the planet. Gates has been funding the ID2020 'alliance' to do just that in league with other lovely people at Microsoft, GAVI, the Rockefeller Foundation, Accenture and IDEO.org. He said in interviews in March, 2020, before any 'vaccine' publicly existed, that the world must have a globalised digital certificate to track the 'virus' and who had been vaccinated. Gates knew from the start that the mRNA vaccines were coming and when they would come and that the plan was to tag the 'vaccinated' to marginalise the intelligent and stop them doing anything including travel. Evil just doesn't suffice. Gates was exposed for offering a $10 million bribe to the Nigerian House of Representatives to invoke compulsory 'Covid' vaccination of all Nigerians. Sara Cunial, a member of the Italian Parliament, called Gates a 'vaccine criminal'. She urged the Italian President to hand him over to the International Criminal Court for crimes against

humanity and condemned his plans to 'chip the human race' through ID2020.

You know it's a long-planned agenda when war criminal and Cult gofer Tony Blair is on the case. With the scale of arrogance only someone as dark as Blair can muster he said: 'Vaccination in the end is going to be your route to liberty.' Blair is a disgusting piece of work and he confirms that again. The media has given a lot of coverage to a bloke called Charlie Mullins, founder of London's biggest independent plumbing company, Pimlico Plumbers, who has said he won't employ anyone who has not been vaccinated or have them go to any home where people are not vaccinated. He said that if he had his way no one would be allowed to walk the streets if they have not been vaccinated. Gates was cheering at the time while I was alerting the white coats. The plan is that people will qualify for 'passports' for having the first two doses and then to keep it they will have to have all the follow ups and new ones for invented 'variants' until human genetics is transformed and many are dead who can't adjust to the changes. Hollywood celebrities – the usual propaganda stunt – are promoting something called the WELL Health-Safety Rating to verify that a building or space has 'taken the necessary steps to prioritize the health and safety of their staff, visitors and other stakeholders'. They included Lady Gaga, Jennifer Lopez, Michael B. Jordan, Robert DeNiro, Venus Williams, Wolfgang Puck, Deepak Chopra and 17th Surgeon General Richard Carmona. Yawn. WELL Health-Safety has big connections with China. Parent company Delos is headed by former Goldman Sachs partner Paul Scialla. This is another example – and we will see so many others – of using the excuse of 'health' to dictate the lives and activities of the population. I guess one confirmation of the 'safety' of buildings is that only 'vaccinated' people can go in, right?

Electronic concentration camps

I wrote decades ago about the plans to restrict travel and here we are for those who refuse to bow to tyranny. This can be achieved in one go with air travel if the aviation industry makes a blanket decree.

The 'vaccine' and guaranteed income are designed to be part of a global version of China's social credit system which tracks behaviour 24/7 and awards or deletes 'credits' based on whether your behaviour is supported by the state or not. I mean your entire lifestyle – what you do, eat, say, everything. Once your credit score falls below a certain level consequences kick in. In China tens of millions have been denied travel by air and train because of this. All the locations and activities denied to refusers by the 'vaccine' passports will be included in one big mass ban on doing almost anything for those that don't bow their head to government. It's beyond fascist and a new term is required to describe its extremes – I guess fascist technocracy will have to do. The way the Chinese system of technological – technocratic – control is sweeping the West can be seen in the Los Angeles school system and is planned to be expanded worldwide. Every child is required to have a 'Covid'-tracking app scanned daily before they can enter the classroom. The so-called Daily Pass tracking system is produced by Gates' Microsoft which I'm sure will shock you rigid. The pass will be scanned using a barcode (one step from an inside-the-body barcode) and the information will include health checks, 'Covid' tests and vaccinations. Entry codes are for one specific building only and access will only be allowed if a student or teacher has a negative test with a test not testing for the 'virus', has no symptoms of anything alleged to be related to 'Covid' (symptoms from a range of other illness), and has a temperature under 100 degrees. No barcode, no entry, is planned to be the case for everywhere and not only schools.

Kids are being psychologically prepared to accept this as 'normal' their whole life which is why what they can impose in schools is so important to the Cult and its gofers. Long-time American freedom campaigner John Whitehead of the Rutherford Institute was not exaggerating when he said: 'Databit by databit, we are building our own electronic concentration camps.' Canada under its Cult gofer prime minister Justin Trudeau has taken a major step towards the real thing with people interned against their will if they test positive with a test not testing for the 'virus' when they arrive at a Canadian

airport. They are jailed in internment hotels often without food or water for long periods and with many doors failing to lock there have been sexual assaults. The interned are being charged sometimes $2,000 for the privilege of being abused in this way. Trudeau is fully on board with the Cult and says the 'Covid pandemic' has provided an opportunity for a global 'reset' to permanently change Western civilisation. His number two, Deputy Prime Minister Chrystia Freeland, is a trustee of the World Economic Forum and a Rhodes Scholar. The Trudeau family have long been servants of the Cult. See *The Biggest Secret* and Cathy O'Brien's book *Trance-Formation of America* for the horrific background to Trudeau's father Pierre Trudeau another Canadian prime minister. Hide your fascism behind the façade of a heart-on-the-sleeve liberal. It's a well-honed Cult technique.

What can the 'vaccine' *really* do?

We have a 'virus' never shown to exist and 'variants' of the 'virus' that have also never been shown to exist except, like the 'original', as computer-generated fictions. Even if you believe there's a 'virus' the 'case' to 'death' rate is in the region of 0.23 to 0.15 percent and those 'deaths' are concentrated among the very old around the same average age that people die anyway. In response to this lack of threat (in truth none) psychopaths and idiots, knowingly and unknowingly answering to Gates and the Cult, are seeking to 'vaccinate' every man, woman and child on Planet Earth. Clearly the 'vaccine' is not about 'Covid' – none of this ever has been. So what is it all about *really*? Why the desperation to infuse genetically-manipulating synthetic material into everyone through mRNA fraudulent 'vaccines' with the intent of doing this over and over with the excuses of 'variants' and other 'virus' inventions? Dr Sherri Tenpenny, an osteopathic medical doctor in the United States, has made herself an expert on vaccines and their effects as a vehement campaigner against their use. Tenpenny was board certified in emergency medicine, the director of a level two trauma centre for 12 years, and moved to Cleveland in 1996 to start an integrative

medicine practice which has treated patients from all 50 states and some 17 other countries. Weaning people off pharmaceutical drugs is a speciality.

She became interested in the consequences of vaccines after attending a meeting at the National Vaccine Information Center in Washington DC in 2000 where she 'sat through four days of listening to medical doctors and scientists and lawyers and parents of vaccine injured kids' and asked: 'What's going on?' She had never been vaccinated and never got ill while her father was given a list of vaccines to be in the military and was 'sick his entire life'. The experience added to her questions and she began to examine vaccine documents from the Centers for Disease Control (CDC). After reading the first one, the 1998 version of *The General Recommendations of Vaccination*, she thought: 'This is it?' The document was poorly written and bad science and Tenpenny began 20 years of research into vaccines that continues to this day. She began her research into 'Covid vaccines' in March, 2020, and she describes them as 'deadly'. For many, as we have seen, they already have been. Tenpenny said that in the first 30 days of the 'vaccine' rollout in the United States there had been more than 40,000 adverse events reported to the vaccine adverse event database. A document had been delivered to her the day before that was 172 pages long. 'We have over 40,000 adverse events; we have over 3,100 cases of [potentially deadly] anaphylactic shock; we have over 5,000 neurological reactions.' Effects ranged from headaches to numbness, dizziness and vertigo, to losing feeling in hands or feet and paraesthesia which is when limbs 'fall asleep' and people have the sensation of insects crawling underneath their skin. All this happened in the first 30 days and remember that only about *ten percent* (or far less) of adverse reactions and vaccine-related deaths are estimated to be officially reported. Tenpenny said:

So can you think of one single product in any industry, any industry, for as long as products have been made on the planet that within 30 days we have 40,000 people complaining of side effects that not only is still on the market but … we've got paid actors telling us how great

they are for getting their vaccine. We're offering people $500 if they will just get their vaccine and we've got nurses and doctors going; 'I got the vaccine, I got the vaccine'.

Tenpenny said they were not going to be 'happy dancing folks' when they began to suffer Bell's palsy (facial paralysis), neuropathies, cardiac arrhythmias and autoimmune reactions that kill through a blood disorder. 'They're not going to be so happy, happy then, but we're never going to see pictures of those people' she said. Tenpenny described the 'vaccine' as 'a well-designed killing tool'.

No off-switch

Bad as the initial consequences had been Tenpenny said it would be maybe 14 months before we began to see the 'full ravage' of what is going to happen to the 'Covid vaccinated' with full-out consequences taking anything between two years and 20 years to show. You can understand why when you consider that variations of the 'Covid vaccine' use mRNA (messenger RNA) to in theory activate the immune system to produce protective antibodies without using the actual 'virus'. How can they when it's a computer program and they've never isolated what they claim is the 'real thing'? Instead they use *synthetic* mRNA. They are inoculating synthetic material into the body which through a technique known as the Trojan horse is absorbed into cells to change the nature of DNA. Human DNA is changed by an infusion of messenger RNA and with each new 'vaccine' of this type it is changed even more. Say so and you are banned by Cult Internet platforms. The contempt the contemptuous Mark Zuckerberg has for the truth and human health can be seen in an internal Facebook video leaked to the Project Veritas investigative team in which he said of the 'Covid vaccines': '… I share some caution on this because we just don't know the long term side-effects of basically modifying people's DNA and RNA.' At the same time this disgusting man's Facebook was censoring and banning anyone saying exactly the same. He must go before a Nuremberg trial for crimes against humanity when he *knows* that he

is censoring legitimate concerns and denying the right of informed consent on behalf of the Cult that owns him. People have been killed and damaged by the very 'vaccination' technique he cast doubt on himself when they may not have had the 'vaccine' with access to information that he denied them. The plan is to have at least annual 'Covid vaccinations', add others to deal with invented 'variants', and change all other vaccines into the mRNA system. Pfizer executives told shareholders at a virtual Barclays Global Healthcare Conference in March, 2021, that the public may need a third dose of 'Covid vaccine', plus regular yearly boosters and the company planned to hike prices to milk the profits in a 'significant opportunity for our vaccine'. These are the professional liars, cheats and opportunists who are telling you their 'vaccine' is safe. Given this volume of mRNA planned to be infused into the human body and its ability to then replicate we will have a transformation of human genetics from biological to synthetic biological – exactly the long-time Cult plan for reasons we'll see – and many will die. Sherri Tenpenny said of this replication:

> It's like having an on-button but no off-button and that whole mechanism … they actually give it a name and they call it the Trojan horse mechanism, because it allows that [synthetic] virus and that piece of that [synthetic] virus to get inside of your cells, start to replicate and even get inserted into other parts of your DNA as a Trojan-horse.

Ask the overwhelming majority of people who have the 'vaccine' what they know about the contents and what they do and they would reply: 'The government says it will stop me getting the virus.' Governments give that false impression on purpose to increase take-up. You can read Sherri Tenpenny's detailed analysis of the health consequences in her blog at Vaxxter.com, but in summary these are some of them. She highlights the statement by Bill Gates about how human beings can become their own 'vaccine manufacturing machine'. The man is insane. ['Vaccine'-generated] 'antibodies' carry synthetic messenger RNA into the cells and the damage starts, Tenpenny contends, and she says that lungs can be adversely affected through varying degrees of pus and bleeding which

obviously affects breathing and would be dubbed 'Covid-19'. Even more sinister was the impact of 'antibodies' on macrophages, a white blood cell of the immune system. They consist of Type 1 and Type 2 which have very different functions. She said Type 1 are 'hyper-vigilant' white blood cells which 'gobble up' bacteria etc. However, in doing so, this could cause inflammation and in extreme circumstances be fatal. She says these affects are mitigated by Type 2 macrophages which kick in to calm down the system and stop it going rogue. They clear up dead tissue debris and reduce inflammation that the Type 1 'fire crews' have caused. Type 1 kills the infection and Type 2 heals the damage, she says. This is her punchline with regard to 'Covid vaccinations': She says that mRNA 'antibodies' block Type 2 macrophages by attaching to them and deactivating them. This meant that when the Type 1 response was triggered by infection there was nothing to stop that getting out of hand by calming everything down. There's an on-switch, but no off-switch, she says. What follows can be 'over and out, see you when I see you'.

Genetic suicide

Tenpenny also highlights the potential for autoimmune disease – the body attacking itself – which has been associated with vaccines since they first appeared. Infusing a synthetic foreign substance into cells could cause the immune system to react in a panic believing that the body is being overwhelmed by an invader (it is) and the consequences can again be fatal. There is an autoimmune response known as a 'cytokine storm' which I have likened to a homeowner panicked by an intruder and picking up a gun to shoot randomly in all directions before turning the fire on himself. The immune system unleashes a storm of inflammatory response called cytokines to a threat and the body commits hara-kiri. The lesson is that you mess with the body's immune response at your peril and these 'vaccines' seriously – fundamentally – mess with immune response. Tenpenny refers to a consequence called anaphylactic shock which is a severe and highly dangerous allergic reaction when the immune system

floods the body with chemicals. She gives the example of having a bee sting which primes the immune system and makes it sensitive to those chemicals. When people are stung again maybe years later the immune response can be so powerful that it leads to anaphylactic shock. Tenpenny relates this 'shock' with regard to the 'Covid vaccine' to something called polyethylene glycol or PEG. Enormous numbers of people have become sensitive to this over decades of use in a whole range of products and processes including food, drink, skin creams and 'medicine'. Studies have claimed that some 72 percent of people have antibodies triggered by PEG compared with two percent in the 1960s and allergic hypersensitive reactions to this become a gathering cause for concern. Tenpenny points out that the 'mRNA vaccine' is coated in a 'bubble' of polyethylene glycol which has the potential to cause anaphylactic shock through immune sensitivity. Many reports have appeared of people reacting this way after having the 'Covid vaccine'. What do we think is going to happen as humanity has more and more of these 'vaccines'? Tenpenny said: 'All these pictures we have seen with people with these rashes … these weepy rashes, big reactions on their arms and things like that – it's an acute allergic reaction most likely to the polyethylene glycol that you've been previously primed and sensitised to.'

Those who have not studied the conspiracy and its perpetrators at length might think that making the population sensitive to PEG and then putting it in these 'vaccines' is just a coincidence. It is not. It is instead testament to how carefully and coldly-planned current events have been and the scale of the conspiracy we are dealing with. Tenpenny further explains that the 'vaccine' mRNA procedure can breach the blood-brain barrier which protects the brain from toxins and other crap that will cause malfunction. In this case they could make two proteins corrupt brain function to cause Amyotrophic lateral sclerosis (ALS) , a progressive nervous system disease leading to loss of muscle control, and frontal lobe degeneration – Alzheimer's and dementia. Immunologist J. Bart Classon published a paper connecting mRNA 'vaccines' to prion

disease which can lead to Alzheimer's and other forms of neurogenerative disease while others have pointed out the potential to affect the placenta in ways that make women infertile. This will become highly significant in the next chapter when I will discuss other aspects of this non-vaccine that relate to its nanotechnology and transmission from the injected to the uninjected.

Qualified in idiocy

Tenpenny describes how research has confirmed that these 'vaccine'-generated antibodies can interact with a range of other tissues in the body and attack many other organs including the lungs. 'This means that if you have a hundred people standing in front of you that all got this shot they could have a hundred different symptoms.' Anyone really think that Cult gofers like the Queen, Tony Blair, Christopher Whitty, Anthony Fauci, and all the other psychopaths have really had this 'vaccine' in the pictures we've seen? Not a bloody chance. Why don't doctors all tell us about all these dangers and consequences of the 'Covid vaccine'? Why instead do they encourage and pressure patients to have the shot? Don't let's think for a moment that doctors and medical staff can't be stupid, lazy, and psychopathic and that's without the financial incentives to give the jab. Tenpenny again:

> Some people are going to die from the vaccine directly but a large number of people are going to start to get horribly sick and get all kinds of autoimmune diseases 42 days to maybe a year out. What are they going to do, these stupid doctors who say; 'Good for you for getting that vaccine.' What are they going to say; 'Oh, it must be a mutant, we need to give an extra dose of that vaccine.'
>
> Because now the vaccine, instead of one dose or two doses we need three or four because the stupid physicians aren't taking the time to learn anything about it. If I can learn this sitting in my living room reading a 19 page paper and several others so can they. There's nothing special about me, I just take the time to do it.

Remember how Sara Kayat, the NHS and TV doctor, said that the 'Covid vaccine' would '100 percent prevent hospitalisation and death'. Doctors can be idiots like every other profession and they

should not be worshipped as infallible. They are not and far from it. Behind many medical and scientific 'experts' lies an uninformed prat trying to hide themselves from you although in the 'Covid' era many have failed to do so as with UK narrative-repeating 'TV doctor' Hilary Jones. Pushing back against the minority of proper doctors and scientists speaking out against the 'vaccine' has been the entire edifice of the Cult global state in the form of governments, medical systems, corporations, mainstream media, Silicon Valley, and an army of compliant doctors, medical staff and scientists willing to say anything for money and to enhance their careers by promoting the party line. If you do that you are an 'expert' and if you won't you are an 'anti-vaxxer' and 'Covidiot'. The pressure to be 'vaccinated' is incessant. We have even had reports claiming that the 'vaccine' can help cure cancer and Alzheimer's and make the lame walk. I am waiting for the announcement that it can bring you coffee in the morning and cook your tea. Just as the symptoms of 'Covid' seem to increase by the week so have the miracles of the 'vaccine'. American supermarket giant Kroger Co. offered nearly 500,000 employees in 35 states a $100 bonus for having the 'vaccine' while donut chain Krispy Kreme promised 'vaccinated' customers a free glazed donut every day for the rest of 2021. Have your DNA changed and you will get a doughnut although we might not have to give you them for long. Such offers and incentives confirm the desperation.

Perhaps the worse vaccine-stunt of them all was UK 'Health' Secretary Matt-the-prat Hancock on live TV after watching a clip of someone being 'vaccinated' when the roll-out began. Hancock faked tears so badly it was embarrassing. Brain-of-Britain Piers Morgan, the lockdown-supporting, 'vaccine' supporting, 'vaccine' passport-supporting, TV host played along with Hancock – 'You're quite emotional about that' he said in response to acting so atrocious it would have been called out at a school nativity which will presumably today include Mary and Jesus in masks, wise men keeping their camels six feet apart, and shepherds under tent arrest. System-serving Morgan tweeted this: 'Love the idea of covid vaccine passports for everywhere: flights, restaurants, clubs, football, gyms,

shops etc. It's time covid-denying, anti-vaxxer loonies had their bullsh*t bluff called & bar themselves from going anywhere that responsible citizens go.' If only I could aspire to his genius. To think that Morgan, who specialises in shouting over anyone he disagrees with, was lauded as a free speech hero when he lost his job after storming off the set of his live show like a child throwing his dolly out of the pram. If he is a free speech hero we are in real trouble. I have no idea what 'bullsh*t' means, by the way, the * throws me completely.

The Cult is desperate to infuse its synthetic DNA-changing concoction into everyone and has been using every lie, trick and intimidation to do so. The question of '*Why*?' we shall now address.

CHAPTER TEN

Human 2.0

I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted –
Alan Turing (1912-1954), the 'Father of artificial intelligence'

I have been exposing for decades the plan to transform the human body from a biological to a synthetic-biological state. The new human that I will call Human 2.0 is planned to be connected to artificial intelligence and a global AI 'Smart Grid' that would operate as one global system in which AI would control everything from your fridge to your heating system to your car to your mind. Humans would no longer be 'human', but post-human and sub-human, with their thinking and emotional processes replaced by AI.

What I said sounded crazy and beyond science fiction and I could understand that. To any balanced, rational, mind it *is* crazy. Today, however, that world is becoming reality and it puts the 'Covid vaccine' into its true context. Ray Kurzweil is the ultra-Zionist 'computer scientist, inventor and futurist' and co-founder of the Singularity University. Singularity refers to the merging of humans with machines or 'transhumanism'. Kurzweil has said humanity would be connected to the cyber 'cloud' in the period of the ever-recurring year of 2030:

Our thinking … will be a hybrid of biological and non-biological thinking … humans will be able to extend their limitations and 'think in the cloud' … We're going to put gateways to the

cloud in our brains ... We're going to gradually merge and enhance ourselves ... In my view, that's the nature of being human – we transcend our limitations. As the technology becomes vastly superior to what we are then the small proportion that is still human gets smaller and smaller and smaller until it's just utterly negligible.

They are trying to sell this end-of-humanity-as-we-know-it as the next stage of 'evolution' when we become super-human and 'like the gods'. They are lying to you. Shocked, eh? The population, and again especially the young, have been manipulated into addiction to technologies designed to enslave them for life. First they induced an addiction to smartphones (holdables); next they moved to technology on the body (wearables); and then began the invasion of the body (implantables). I warned way back about the plan for microchipped people and we are now entering that era. We should not be diverted into thinking that this refers only to chips we can see. Most important are the nanochips known as smart dust, neural dust and nanobots which are far too small to be seen by the human eye. Nanotechnology is everywhere, increasingly in food products, and released into the atmosphere by the geoengineering of the skies funded by Bill Gates to 'shut out the Sun' and 'save the planet from global warming'. Gates has been funding a project to spray millions of tonnes of chalk (calcium carbonate) into the stratosphere over Sweden to 'dim the Sun' and cool the Earth. Scientists warned the move could be disastrous for weather systems in ways no one can predict and opposition led to the Swedish space agency announcing that the 'experiment' would not be happening as planned in the summer of 2021; but it shows where the Cult is going with dimming the impact of the Sun and there's an associated plan to change the planet's atmosphere. Who gives psychopath Gates the right to dictate to the entire human race and dismantle planetary systems? The world will not be safe while this man is at large.

The global warming hoax has made the Sun, like the gas of life, something to fear when both are essential to good health and human survival (more inversion). The body transforms sunlight into vital vitamin D through a process involving … *cholesterol*. This is the cholesterol we are also told to fear. We are urged to take Big Pharma

statin drugs to reduce cholesterol and it's all systematic. Reducing cholesterol means reducing vitamin D uptake with all the multiple health problems that will cause. At least if you take statins long term it saves the government from having to pay you a pension. The delivery system to block sunlight is widely referred to as chemtrails although these have a much deeper agenda, too. They appear at first to be contrails or condensation trails streaming from aircraft into cold air at high altitudes. Contrails disperse very quickly while chemtrails do not and spread out across the sky before eventually their content falls to earth. Many times I have watched aircraft cross-cross a clear blue sky releasing chemtrails until it looks like a cloudy day. Chemtrails contain many things harmful to humans and the natural world including toxic heavy metals, aluminium (see Alzheimer's) and nanotechnology. Ray Kurzweil reveals the reason without actually saying so: 'Nanobots will infuse all the matter around us with information. Rocks, trees, everything will become these intelligent creatures.' How do you deliver that? *From the sky*. Self-replicating nanobots would connect everything to the Smart Grid. The phenomenon of Morgellons disease began in the chemtrail era and the correlation has led to it being dubbed the 'chemtrail disease'. Self-replicating fibres appear in the body that can be pulled out through the skin. Morgellons fibres continue to grow outside the body and have a form of artificial intelligence. I cover this at greater length in *Phantom Self*.

'Vaccine' operating system

'Covid vaccines' with their self-replicating synthetic material are also designed to make the connection between humanity and Kurzweil's 'cloud'. American doctor and dedicated campaigner for truth, Carrie Madej, an Internal Medicine Specialist in Georgia with more than 20 years medical experience, has highlighted the nanotechnology aspect of the fake 'vaccines'. She explains how one of the components in at least the Moderna and Pfizer synthetic potions are 'lipid nanoparticles' which are 'like little tiny computer bits' – a 'sci-fi substance' known as nanobots and hydrogel which can be 'triggered

at any moment to deliver its payload' and act as 'biosensors'. The synthetic substance had 'the ability to accumulate data from your body like your breathing, your respiration, thoughts and emotions, all kind of things' and each syringe could carry a *million* nanobots:

This substance because it's like little bits of computers in your body, crazy, but it's true, it can do that, [and] obviously has the ability to act through Wi-Fi. It can receive and transmit energy, messages, frequencies or impulses. That issue has never been addressed by these companies. What does that do to the human?

Just imagine getting this substance in you and it can react to things all around you, the 5G, your smart device, your phones, what is happening with that? What if something is triggering it, too, like an impulse, a frequency? We have something completely foreign in the human body.

Madej said her research revealed that electromagnetic (EMF) frequencies emitted by phones and other devices had increased dramatically in the same period of the 'vaccine' rollout and she was seeing more people with radiation problems as 5G and other electromagnetic technology was expanded and introduced to schools and hospitals. She said she was 'floored with the EMF coming off' the devices she checked. All this makes total sense and syncs with my own work of decades when you think that Moderna refers in documents to its mRNA 'vaccine' as an 'operating system':

Recognizing the broad potential of mRNA science, we set out to create an mRNA technology platform that functions very much like an operating system on a computer. It is designed so that it can plug and play interchangeably with different programs. In our case, the 'program' or 'app' is our mRNA drug – the unique mRNA sequence that codes for a protein …

… Our MRNA Medicines – 'The 'Software Of Life': When we have a concept for a new mRNA medicine and begin research, fundamental components are already in place. Generally, the only thing that changes from one potential mRNA medicine to another is the coding region – the actual genetic code that instructs ribosomes to make protein. Utilizing these instruction sets gives our investigational mRNA medicines a software-like quality. We also have the ability to combine different mRNA sequences encoding for different proteins in a single mRNA investigational medicine.

Who needs a real 'virus' when you can create a computer version to justify infusing your operating system into the entire human race on the road to making living, breathing people into cyborgs? What is missed with the 'vaccines' is the *digital* connection between synthetic material and the body that I highlighted earlier with the study that hacked a computer with human DNA. On one level the body is digital, based on mathematical codes, and I'll have more about that in the next chapter. Those who ridiculously claim that mRNA 'vaccines' are not designed to change human genetics should explain the words of Dr Tal Zaks, chief medical officer at Moderna, in a 2017 TED talk. He said that over the last 30 years 'we've been living this phenomenal digital scientific revolution, and I'm here today to tell you, that we are actually *hacking the software of life*, and that it's changing the way we think about prevention and treatment of disease':

In every cell there's this thing called messenger RNA, or mRNA for short, that transmits the critical information from the DNA in our genes to the protein, which is really the stuff we're all made out of. This is the critical information that determines what the cell will do. So we think about it as an operating system. So if you could change that, if you could introduce a line of code, or change a line of code, it turns out, that has profound implications for everything, from the flu to cancer.

Zaks should more accurately have said that this has profound implications for the human genetic code and the nature of DNA. Communications within the body go both ways and not only one. But, hey, no, the 'Covid vaccine' will not affect your genetics. Cult fact-checkers say so even though the man who helped to develop the mRNA technique says that it does. Zaks said in 2017:

If you think about what it is we're trying to do. We've taken information and our understanding of that information and how that information is transmitted in a cell, and we've taken our understanding of medicine and how to make drugs, and we're fusing the two. We think of it as information therapy.

I have been writing for decades that the body is an information field communicating with itself and the wider world. This is why

radiation which is information can change the information field of body and mind through phenomena like 5G and change their nature and function. 'Information therapy' means to change the body's information field and change the way it operates. DNA is a receiver-transmitter of information and can be mutated by information like mRNA synthetic messaging. Technology to do this has been ready and waiting in the underground bases and other secret projects to be rolled out when the 'Covid' hoax was played. 'Trials' of such short and irrelevant duration were only for public consumption. When they say the 'vaccine' is 'experimental' that is not true. It may appear to be 'experimental' to those who don't know what's going on, but the trials have already been done to ensure the Cult gets the result it desires. Zaks said that it took decades to sequence the human genome, completed in 2003, but now they could do it in a week. By 'they' he means scientists operating in the public domain. In the secret projects they were sequencing the genome in a week long before even 2003.

Deluge of mRNA

Highly significantly the Moderna document says the guiding premise is that if using mRNA as a medicine works for one disease then it should work for many diseases. They were leveraging the flexibility afforded by their platform and the fundamental role mRNA plays in protein synthesis to pursue mRNA medicines for a broad spectrum of diseases. Moderna is confirming what I was saying through 2020 that multiple 'vaccines' were planned for 'Covid' (and later invented 'variants') and that previous vaccines would be converted to the mRNA system to infuse the body with massive amounts of genetically-manipulating synthetic material to secure a transformation to a synthetic-biological state. The 'vaccines' are designed to kill stunning numbers as part of the long-exposed Cult depopulation agenda and transform the rest. Given this is the goal you can appreciate why there is such hysterical demand for every human to be 'vaccinated' for an alleged 'disease' that has an estimated 'infection' to 'death' ratio of 0.23-0.15 percent. As I write

children are being given the 'vaccine' in trials (their parents are a disgrace) and ever-younger people are being offered the vaccine for a 'virus' that even if you believe it exists has virtually zero chance of harming them. Horrific effects of the 'trials' on a 12-year-old girl were revealed by a family member to be serious brain and gastric problems that included a bowel obstruction and the inability to swallow liquids or solids. She was unable to eat or drink without throwing up, had extreme pain in her back, neck and abdomen, and was paralysed from the waist down which stopped her urinating unaided. When the girl was first taken to hospital doctors said it was all in her mind. She was signed up for the 'trial' by her parents for whom no words suffice. None of this 'Covid vaccine' insanity makes any sense unless you see what the 'vaccine' really is – a body-changer. Synthetic biology or 'SynBio' is a fast-emerging and expanding scientific discipline which includes everything from genetic and molecular engineering to electrical and computer engineering. Synthetic biology is defined in these ways:

- A multidisciplinary area of research that seeks to create new biological parts, devices, and systems, or to redesign systems that are already found in nature.

- The use of a mixture of physical engineering and genetic engineering to create new (and therefore synthetic) life forms.

- An emerging field of research that aims to combine the knowledge and methods of biology, engineering and related disciplines in the design of chemically-synthesized DNA to create organisms with novel or enhanced characteristics and traits (synthetic organisms including humans).

We now have synthetic blood, skin, organs and limbs being developed along with synthetic body parts produced by 3D printers. These are all elements of the synthetic human programme and this comment by Kurzweil's co-founder of the Singularity University,

Peter Diamandis, can be seen in a whole new light with the 'Covid' hoax and the sanctions against those that refuse the 'vaccine':

Anybody who is going to be resisting the progress forward [to transhumanism] is going to be resisting evolution and, fundamentally, they will die out. It's not a matter of whether it's good or bad. It's going to happen.

'Resisting evolution'? What absolute bollocks. The arrogance of these people is without limit. His 'it's going to happen' mantra is another way of saying 'resistance is futile' to break the spirit of those pushing back and we must not fall for it. Getting this genetically-transforming 'vaccine' into everyone is crucial to the Cult plan for total control and the desperation to achieve that is clear for anyone to see. Vaccine passports are a major factor in this and they, too, are a form of resistance is futile. It's NOT. The paper funded by the Rockefeller Foundation for the 2013 'health conference' in China said:

We will interact more with artificial intelligence. The use of robotics, bio-engineering to augment human functioning is already well underway and will advance. Re-engineering of humans into potentially separate and unequal forms through genetic engineering or mixed human-robots raises debates on ethics and equality.

A new demography is projected to emerge after 2030 [that year again] of technologies (robotics, genetic engineering, nanotechnology) producing robots, engineered organisms, 'nanobots' and artificial intelligence (AI) that can self-replicate. Debates will grow on the implications of an impending reality of human designed life.

What is happening today is so long planned. The world army enforcing the will of the world government is intended to be a robot army, not a human one. Today's military and its technologically 'enhanced' troops, pilotless planes and driverless vehicles are just stepping stones to that end. Human soldiers are used as Cult fodder and its time they woke up to that and worked for the freedom of the population instead of their own destruction and their family's destruction – the same with the police. Join us and let's sort this out. The phenomenon of enforce my own destruction is widespread in the 'Covid' era with Woker 'luvvies' in the acting and entertainment

industries supporting 'Covid' rules which have destroyed their profession and the same with those among the public who put signs on the doors of their businesses 'closed due to Covid – stay safe' when many will never reopen. It's a form of masochism and most certainly insanity.

Transgender = transhumanism

When something explodes out of nowhere and is suddenly everywhere it is always the Cult agenda and so it is with the tidal wave of claims and demands that have infiltrated every aspect of society under the heading of 'transgenderism'. The term 'trans' is so 'in' and this is the dictionary definition:

> A prefix meaning 'across', 'through', occurring … in loanwords from Latin, used in particular for denoting movement or conveyance from place to place (transfer; transmit; transplant) or complete change (transform; transmute), or to form adjectives meaning 'crossing', 'on the other side of', or 'going beyond' the place named (transmontane; transnational; trans-Siberian).

Transgender means to go beyond gender and transhuman means to go beyond human. Both are aspects of the Cult plan to transform the human body to a synthetic state with *no gender*. Human 2.0 is not designed to procreate and would be produced technologically with no need for parents. The new human would mean the end of parents and so men, and increasingly women, are being targeted for the deletion of their rights and status. Parental rights are disappearing at an ever-quickening speed for the same reason. The new human would have no need for men or women when there is no procreation and no gender. Perhaps the transgender movement that appears to be in a permanent state of frenzy might now contemplate on how it is being used. This was never about transgender rights which are only the interim excuse for confusing gender, particularly in the young, on the road to *fusing* gender. Transgender activism is not an end; it is a *means* to an end. We see again the technique of creative destruction in which you destroy the status quo to 'build back better' in the form that you want. The gender status quo had to be

destroyed by persuading the Cult-created Woke mentality to believe that you can have 100 genders or more. A programme for 9 to 12 year olds produced by the Cult-owned BBC promoted the 100 genders narrative. The very idea may be the most monumental nonsense, but it is not what is true that counts, only what you can make people *believe* is true. Once the gender of 2 + 2 = 4 has been dismantled through indoctrination, intimidation and 2 + 2 = 5 then the new no-gender normal can take its place with Human 2.0. Aldous Huxley revealed the plan in his prophetic *Brave New World* in 1932:

> Natural reproduction has been done away with and children are created, decanted', and raised in 'hatcheries and conditioning centres'. From birth, people are genetically designed to fit into one of five castes, which are further split into 'Plus' and 'Minus' members and designed to fulfil predetermined positions within the social and economic strata of the World State.

How could Huxley know this in 1932? For the same reason George Orwell knew about the Big Brother state in 1948, Cult insiders I have quoted knew about it in 1969, and I have known about it since the early 1990s. If you are connected to the Cult or you work your balls off to uncover the plan you can predict the future. The process is simple. If there is a plan for the world and nothing intervenes to stop it then it will happen. Thus if you communicate the plan ahead of time you are perceived to have predicted the future, but you haven't. You have revealed the plan which without intervention will become the human future. The whole reason I have done what I have is to alert enough people to inspire an intervention and maybe at last that time has come with the Cult and its intentions now so obvious to anyone with a brain in working order.

The future is here

Technological wombs that Huxley described to replace parent procreation are already being developed and they are only the projects we know about in the public arena. Israeli scientists told *The Times of Israel* in March, 2021, that they have grown 250-cell embryos

into mouse foetuses with fully formed organs using artificial wombs in a development they say could pave the way for gestating humans outside the womb. Professor Jacob Hanna of the Weizmann Institute of Science said:

> We took mouse embryos from the mother at day five of development, when they are just of 250 cells, and had them in the incubator from day five until day 11, by which point they had grown all their organs.
>
> By day 11 they make their own blood and have a beating heart, a fully developed brain. Anybody would look at them and say, 'this is clearly a mouse foetus with all the characteristics of a mouse.' It's gone from being a ball of cells to being an advanced foetus.

A special liquid is used to nourish embryo cells in a laboratory dish and they float on the liquid to duplicate the first stage of embryonic development. The incubator creates all the right conditions for its development, Hanna said. The liquid gives the embryo 'all the nutrients, hormones and sugars they need' along with a custom-made electronic incubator which controls gas concentration, pressure and temperature. The cutting-edge in the underground bases and other secret locations will be light years ahead of that, however, and this was reported by the London *Guardian* in 2017:

> We are approaching a biotechnological breakthrough. Ectogenesis, the invention of a complete external womb, could completely change the nature of human reproduction. In April this year, researchers at the Children's Hospital of Philadelphia announced their development of an artificial womb.

The article was headed 'Artificial wombs could soon be a reality. What will this mean for women?' What would it mean for children is an even bigger question. No mother to bond with only a machine in preparation for a life of soulless interaction and control in a world governed by machines (see the *Matrix* movies). Now observe the calculated manipulations of the 'Covid' hoax as human interaction and warmth has been curtailed by distancing, isolation and fear with people communicating via machines on a scale never seen before.

These are all dots in the same picture as are all the personal assistants, gadgets and children's toys through which kids and adults communicate with AI as if it is human. The AI 'voice' on Sat-Nav should be included. All these things are psychological preparation for the Cult endgame. Before you can make a physical connection with AI you have to make a psychological connection and that is what people are being conditioned to do with this ever gathering human-AI interaction. Movies and TV programmes depicting the transhuman, robot dystopia relate to a phenomenon known as 'pre-emptive programming' in which the world that is planned is portrayed everywhere in movies, TV and advertising. This is conditioning the conscious and subconscious mind to become familiar with the planned reality to dilute resistance when it happens for real. What would have been a shock such is the change is made less so. We have young children put on the road to transgender transition surgery with puberty blocking drugs at an age when they could never be able to make those life-changing decisions.

Rachel Levine, a professor of paediatrics and psychiatry who believes in treating children this way, became America's highest-ranked openly-transgender official when she was confirmed as US Assistant Secretary at the Department of Health and Human Services after being nominated by Joe Biden (the Cult). Activists and governments press for laws to deny parents a say in their children's transition process so the kids can be isolated and manipulated into agreeing to irreversible medical procedures. A Canadian father Robert Hoogland was denied bail by the Vancouver Supreme Court in 2021 and remained in jail for breaching a court order that he stay silent over his young teenage daughter, a minor, who was being offered life-changing hormone therapy without parental consent. At the age of 12 the girl's 'school counsellor' said she may be transgender, referred her to a doctor and told the school to treat her like a boy. This is another example of state-serving schools imposing ever more control over children's lives while parents have ever less.

Contemptible and extreme child abuse is happening all over the world as the Cult gender-fusion operation goes into warp-speed.

Why the war on men – and now women?

The question about what artificial wombs mean for women should rightly be asked. The answer can be seen in the deletion of women's rights involving sport, changing rooms, toilets and status in favour of people in male bodies claiming to identify as women. I can identify as a mountain climber, but it doesn't mean I can climb a mountain any more than a biological man can be a biological woman. To believe so is a triumph of belief over factual reality which is the very perceptual basis of everything Woke. Women's sport is being destroyed by allowing those with male bodies who say they identify as female to 'compete' with girls and women. Male body 'women' dominate 'women's' competition with their greater muscle mass, bone density, strength and speed. With that disadvantage sport for women loses all meaning. To put this in perspective nearly 300 American high school boys can run faster than the quickest woman sprinter in the world. Women are seeing their previously protected spaces invaded by male bodies simply because they claim to identify as women. That's all they need to do to access all women's spaces and activities under the Biden 'Equality Act' that destroys equality for women with the usual Orwellian Woke inversion. Male sex offenders have already committed rapes in women's prisons after claiming to identify as women to get them transferred. Does this not matter to the Woke 'equality' hypocrites? Not in the least. What matters to Cult manipulators and funders behind transgender activists is to advance gender fusion on the way to the no-gender 'human'. When you are seeking to impose transparent nonsense like this, or the 'Covid' hoax, the only way the nonsense can prevail is through censorship and intimidation of dissenters, deletion of factual information, and programming of the unquestioning, bewildered and naive. You don't have to scan the world for long to see that all these things are happening.

Many women's rights organisations have realised that rights and status which took such a long time to secure are being eroded and that it is systematic. Kara Dansky of the global Women's Human Rights Campaign said that Biden's transgender executive order immediately he took office, subsequent orders, and Equality Act legislation that followed 'seek to erase women and girls in the law as a category'. *Exactly*. I said during the long ago-started war on men (in which many women play a crucial part) that this was going to turn into a war on them. The Cult is phasing out *both* male and female genders. To get away with that they are brought into conflict so they are busy fighting each other while the Cult completes the job with no unity of response. Unity, people, *unity*. We need unity everywhere. Transgender is the only show in town as the big step towards the no-gender human. It's not about rights for transgender people and never has been. Woke political correctness is deleting words relating to genders to the same end. Wokers believe this is to be 'inclusive' when the opposite is true. They are deleting words describing gender because gender *itself* is being deleted by Human 2.0. Terms like 'man', 'woman', 'mother' and 'father' are being deleted in the universities and other institutions to be replaced by the *no*-gender, not trans-gender, 'individuals' and 'guardians'. Women's rights campaigner Maria Keffler of Partners for Ethical Care said: 'Children are being taught from kindergarten upward that some boys have a vagina, some girls have a penis, and that kids can be any gender they want to be.' Do we really believe that suddenly countries all over the world at the same time had the idea of having drag queens go into schools or read transgender stories to very young children in the local library? It's coldly-calculated confusion of gender on the way to the fusion of gender. Suzanne Vierling, a psychologist from Southern California, made another important point:

> Yesterday's slave woman who endured gynecological medical experiments is today's girl-child being butchered in a booming gender-transitioning sector. Ovaries removed, pushing her into menopause and osteoporosis, uncharted territory, and parents' rights and authority decimated.

The erosion of parental rights is a common theme in line with the Cult plans to erase the very concept of parents and 'ovaries removed, pushing her into menopause' means what? Those born female lose the ability to have children – another way to discontinue humanity as we know it.

Eliminating Human 1.0 (before our very eyes)

To pave the way for Human 2.0 you must phase out Human 1.0. This is happening through plummeting sperm counts and making women infertile through an onslaught of chemicals, radiation (including smartphones in pockets of men) and mRNA 'vaccines'. Common agriculture pesticides are also having a devastating impact on human fertility. I have been tracking collapsing sperm counts in the books for a long time and in 2021 came a book by fertility scientist and reproductive epidemiologist Shanna Swan, *Count Down: How Our Modern World Is Threatening Sperm Counts, Altering Male and Female Reproductive Development and Imperiling the Future of the Human Race*. She reports how the global fertility rate dropped by *half* between 1960 and 2016 with America's birth rate 16 percent below where it needs to be to sustain the population. Women are experiencing declining egg quality, more miscarriages, and more couples suffer from infertility. Other findings were an increase in erectile dysfunction, infant boys developing more genital abnormalities, male problems with conception, and plunging levels of the male hormone testosterone which would explain why so many men have lost their backbone and masculinity. This has been very evident during the 'Covid' hoax when women have been prominent among the Pushbackers and big strapping blokes have bowed their heads, covered their faces with a nappy and quietly submitted. Mind control expert Cathy O'Brien also points to how global education introduced the concept of 'we're all winners' in sport and classrooms: 'Competition was defused, and it in turn defused a sense of fighting back.' This is another version of the 'equity' doctrine in which you drive down rather than raise up. What a contrast in Cult-controlled China with its global ambitions

where the government published plans in January, 2021, to 'cultivate masculinity' in boys from kindergarten through to high school in the face of a 'masculinity crisis'. A government adviser said boys would be soon become 'delicate, timid and effeminate' unless action was taken. Don't expect any similar policy in the targeted West. A 2006 study showed that a 65-year-old man in 2002 had testosterone levels *15 percent* lower than a 65-year-old man in 1987 while a 2020 study found a similar story with young adults and adolescents. Men are getting prescriptions for testosterone replacement therapy which causes an even greater drop in sperm count with up to 99 percent seeing sperm counts drop to zero during the treatment. More sperm is defective and malfunctioning with some having two heads or not pursuing an egg.

A class of *synthetic* chemicals known as phthalates are being blamed for the decline. These are found everywhere in plastics, shampoos, cosmetics, furniture, flame retardants, personal care products, pesticides, canned foods and even receipts. Why till receipts? Everyone touches them. Let no one delude themselves that all this is not systematic to advance the long-time agenda for human body transformation. Phthalates mimic hormones and disrupt the hormone balance causing testosterone to fall and genital birth defects in male infants. Animals and fish have been affected in the same way due to phthalates and other toxins in rivers. When fish turn gay or change sex through chemicals in rivers and streams it is a pointer to why there has been such an increase in gay people and the sexually confused. It doesn't matter to me what sexuality people choose to be, but if it's being affected by chemical pollution and consumption then we need to know. Does anyone really think that this is not connected to the transgender agenda, the war on men and the condemnation of male 'toxic masculinity'? You watch this being followed by 'toxic femininity'. It's already happening. When breastfeeding becomes 'chest-feeding', pregnant women become pregnant people along with all the other Woke claptrap you know that the world is going insane and there's a Cult scam in progress. Transgender activists are promoting the Cult agenda while Cult

billionaires support and fund the insanity as they laugh themselves to sleep at the sheer stupidity for which humans must be infamous in galaxies far, far away.

'Covid vaccines' and female infertility

We can now see why the 'vaccine' has been connected to potential infertility in women. Dr Michael Yeadon, former Vice President and Chief Scientific Advisor at Pfizer, and Dr Wolfgang Wodarg in Germany, filed a petition with the European Medicines Agency in December, 2020, urging them to stop trials for the Pfizer/BioNTech shot and all other mRNA trials until further studies had been done. They were particularly concerned about possible effects on fertility with 'vaccine'-produced antibodies attacking the protein Syncytin-1 which is responsible for developing the placenta. The result would be infertility 'of indefinite duration' in women who have the 'vaccine' with the placenta failing to form. Section 10.4.2 of the Pfizer/BioNTech trial protocol says that pregnant women or those who might become so should not have mRNA shots. Section 10.4 warns men taking mRNA shots to 'be abstinent from heterosexual intercourse' and not to donate sperm. The UK government said that it *did not know* if the mRNA procedure had an effect on fertility. *Did not know?* These people have to go to jail. UK government advice did not recommend at the start that pregnant women had the shot and said they should avoid pregnancy for at least two months after 'vaccination'. The 'advice' was later updated to pregnant women should only have the 'vaccine' if the benefits outweighed the risks to mother and foetus. What the hell is that supposed to mean? Then 'spontaneous abortions' began to appear and rapidly increase on the adverse reaction reporting schemes which include only a fraction of adverse reactions. Thousands and ever-growing numbers of 'vaccinated' women are describing changes to their menstrual cycle with heavier blood flow, irregular periods and menstruating again after going through the menopause – all links to reproduction effects. Women are passing blood clots and the lining of their uterus while men report erectile dysfunction and blood effects. Most

significantly of all *un*vaccinated women began to report similar menstrual changes after interaction with *'vaccinated'* people and men and children were also affected with bleeding noses, blood clots and other conditions. 'Shedding' is when vaccinated people can emit the content of a vaccine to affect the unvaccinated, but this is different. 'Vaccinated' people were not shedding a 'live virus' allegedly in 'vaccines' as before because the fake 'Covid vaccines' involve synthetic material and other toxicity. Doctors exposing what is happening prefer the term 'transmission' to shedding. Somehow those that have had the shots are transmitting effects to those that haven't. Dr Carrie Madej said the nano-content of the 'vaccines' can 'act like an antenna' to others around them which fits perfectly with my own conclusions. This 'vaccine' transmission phenomenon was becoming known as the book went into production and I deal with this further in the Postscript.

Vaccine effects on sterility are well known. The World Health Organization was accused in 2014 of sterilising millions of women in Kenya with the evidence confirmed by the content of the vaccines involved. The same WHO behind the 'Covid' hoax admitted its involvement for more than ten years with the vaccine programme. Other countries made similar claims. Charges were lodged by Tanzania, Nicaragua, Mexico, and the Philippines. The Gardasil vaccine claimed to protect against a genital 'virus' known as HPV has also been linked to infertility. Big Pharma and the WHO (same thing) are criminal and satanic entities. Then there's the Bill Gates Foundation which is connected through funding and shared interests with 20 pharmaceutical giants and laboratories. He stands accused of directing the policy of United Nations Children's Fund (UNICEF), vaccine alliance GAVI, and other groupings, to advance the vaccine agenda and silence opposition at great cost to women and children. At the same time Gates wants to reduce the global population. Coincidence?

Great Reset = Smart Grid = new human

The Cult agenda I have been exposing for 30 years is now being openly promoted by Cult assets like Gates and Klaus Schwab of the World Economic Forum under code-terms like the 'Great Reset', 'Build Back Better' and 'a rare but narrow window of opportunity to reflect, reimagine, and reset our world'. What provided this 'rare but narrow window of opportunity'? The 'Covid' hoax did. Who created that? *They* did. My books from not that long ago warned about the planned 'Internet of Things' (IoT) and its implications for human freedom. This was the plan to connect all technology to the Internet and artificial intelligence and today we are way down that road with an estimated 36 billion devices connected to the World Wide Web and that figure is projected to be 76 billion by 2025. I further warned that the Cult planned to go beyond that to the Internet of *Everything* when the human brain was connected via AI to the Internet and Kurzweil's 'cloud'. Now we have Cult operatives like Schwab calling for precisely that under the term 'Internet of Bodies', a fusion of the physical, digital and biological into one centrally-controlled Smart Grid system which the Cult refers to as the 'Fourth Industrial Revolution'. They talk about the 'biological', but they really mean the synthetic-biological which is required to fully integrate the human body and brain into the Smart Grid and artificial intelligence planned to replace the human mind. We have everything being synthetically manipulated including the natural world through GMO and smart dust, the food we eat and the human body itself with synthetic 'vaccines'. I said in *The Answer* that we would see the Cult push for synthetic meat to replace animals and in February, 2021, the so predictable psychopath Bill Gates called for the introduction of synthetic meat to save us all from 'climate change'. The climate hoax just keeps on giving like the 'Covid' hoax. The war on meat by vegan activists is a carbon (oops, sorry) copy of the manipulation of transgender activists. They have no idea (except their inner core) that they are being used to promote and impose the agenda of the Cult or that they are only the *vehicle* and not the *reason*. This is not to say those who choose not to eat meat shouldn't be respected and supported in that right, but there are ulterior motives

for those in power. A *Forbes* article in December, 2019, highlighted the plan so beloved of Schwab and the Cult under the heading: 'What Is The Internet of Bodies? And How Is It Changing Our World?' The article said the human body is the latest data platform (remember 'our vaccine is an operating system'). *Forbes* described the plan very accurately and the words could have come straight out of my books from long before:

> The Internet of Bodies (IoB) is an extension of the IoT and basically connects the human body to a network through devices that are ingested, implanted, or connected to the body in some way. Once connected, data can be exchanged, and the body and device can be remotely monitored and controlled.

They were really describing a human hive mind with human perception centrally-dictated via an AI connection as well as allowing people to be 'remotely monitored and controlled'. Everything from a fridge to a human mind could be directed from a central point by these insane psychopaths and 'Covid vaccines' are crucial to this. Forbes explained the process I mentioned earlier of holdable and wearable technology followed by implantable. The article said there were three generations of the Internet of Bodies that include:

- Body external: These are wearable devices such as Apple Watches or Fitbits that can monitor our health.

- Body internal: These include pacemakers, cochlear implants, and digital pills that go inside our bodies to monitor or control various aspects of health.

- Body embedded: The third generation of the Internet of Bodies is embedded technology where technology and the human body are melded together and have a real-time connection to a remote machine.

Forbes noted the development of the Brain Computer Interface (BCI) which merges the brain with an external device for monitoring and controlling in real-time. 'The ultimate goal is to help restore function to individuals with disabilities by using brain signals rather than conventional neuromuscular pathways.' Oh, do fuck off. The goal of brain interface technology is controlling human thought and emotion from the central point in a hive mind serving its masters wishes. Many people are now agreeing to be chipped to open doors without a key. You can recognise them because they'll be wearing a mask, social distancing and lining up for the 'vaccine'. The Cult plans a Great Reset money system after they have completed the demolition of the global economy in which 'money' will be exchanged through communication with body operating systems. Rand Corporation, a Cult-owned think tank, said of the Internet of Bodies or IoB:

> Internet of Bodies technologies fall under the broader IoT umbrella. But as the name suggests, IoB devices introduce an even more intimate interplay between humans and gadgets. IoB devices monitor the human body, collect health metrics and other personal information, and transmit those data over the Internet. Many devices, such as fitness trackers, are already in use … IoB devices … and those in development can track, record, and store users' whereabouts, bodily functions, and what they see, hear, and even think.

Schwab's World Economic Forum, a long-winded way of saying 'fascism' or 'the Cult', has gone full-on with the Internet of Bodies in the 'Covid' era. 'We're entering the era of the Internet of Bodies', it declared, 'collecting our physical data via a range of devices that can be implanted, swallowed or worn'. The result would be a huge amount of health-related data that could improve human wellbeing around the world, and prove crucial in fighting the 'Covid-19 pandemic'. Does anyone think these clowns care about 'human wellbeing' after the death and devastation their pandemic hoax has purposely caused? Schwab and co say we should move forward with the Internet of Bodies because 'Keeping track of symptoms could help us stop the spread of infection, and quickly detect new cases'. How wonderful, but keeping track' is all they are really bothered

about. Researchers were investigating if data gathered from smartwatches and similar devices could be used as viral infection alerts by tracking the user's heart rate and breathing. Schwab said in his 2018 book *Shaping the Future of the Fourth Industrial Revolution*:

> The lines between technologies and beings are becoming blurred and not just by the ability to create lifelike robots or synthetics. Instead it is about the ability of new technologies to literally become part of us. Technologies already influence how we understand ourselves, how we think about each other, and how we determine our realities. As the technologies … give us deeper access to parts of ourselves, we may begin to integrate digital technologies into our bodies.

You can see what the game is. Twenty-four hour control and people – if you could still call them that – would never know when something would go ping and take them out of circulation. It's the most obvious rush to a global fascist dictatorship and the complete submission of humanity and yet still so many are locked away in their Cult-induced perceptual coma and can't see it.

Smart Grid control centres

The human body is being transformed by the 'vaccines' and in other ways into a synthetic cyborg that can be attached to the global Smart Grid which would be controlled from a central point and other sub-locations of Grid manipulation. Where are these planned to be? Well, China for a start which is one of the Cult's biggest centres of operation. The technological control system and technocratic rule was incubated here to be unleashed across the world after the 'Covid' hoax came out of China in 2020. Another Smart Grid location that will surprise people new to this is Israel. I have exposed in *The Trigger* how Sabbatian technocrats, intelligence and military operatives were behind the horrors of 9/11 and not '19 Arab hijackers' who somehow manifested the ability to pilot big passenger airliners when instructors at puddle-jumping flying schools described some of them as a joke. The 9/11 attacks were made possible through control of civilian and military air computer systems and those of the White House, Pentagon and connected agencies. See *The Trigger* – it

will blow your mind. The controlling and coordinating force were the Sabbatian networks in Israel and the United States which by then had infiltrated the entire US government, military and intelligence system. The real name of the American Deep State is 'Sabbatian State'. Israel is a tiny country of only nine million people, but it is one of the global centres of cyber operations and fast catching Silicon Valley in importance to the Cult. Israel is known as the 'start-up nation' for all the cyber companies spawned there with the Sabbatian specialisation of 'cyber security' that I mentioned earlier which gives those companies access to computer systems of their clients in real time through 'backdoors' written into the coding when security software is downloaded. The Sabbatian centre of cyber operations outside Silicon Valley is the Israeli military Cyber Intelligence Unit, the biggest infrastructure project in Israel's history, headquartered in the desert-city of Beersheba and involving some 20,000 'cyber soldiers'. Here are located a literal army of Internet trolls scanning social media, forums and comment lists for anyone challenging the Cult agenda. The UK military has something similar with its 77th Brigade and associated operations. The Beersheba complex includes research and development centres for other Cult operations such as Intel, Microsoft, IBM, Google, Apple, Hewlett-Packard, Cisco Systems, Facebook and Motorola. Techcrunch.com ran an article about the Beersheba global Internet technology centre headlined 'Israel's desert city of Beersheba is turning into a cybertech oasis':

> The military's massive relocation of its prestigious technology units, the presence of multinational and local companies, a close proximity to Ben Gurion University and generous government subsidies are turning Beersheba into a major global cybertech hub. Beersheba has all of the ingredients of a vibrant security technology ecosystem, including Ben Gurion University with its graduate program in cybersecurity and Cyber Security Research Center, and the presence of companies such as EMC, Deutsche Telekom, PayPal, Oracle, IBM, and Lockheed Martin. It's also the future home of the INCB (Israeli National Cyber Bureau); offers a special income tax incentive for cyber security companies, and was the site for the relocation of the army's intelligence corps units.

Sabbatians have taken over the cyber world through the following process: They scan the schools for likely cyber talent and develop them at Ben Gurion University and their period of conscription in the Israeli Defense Forces when they are stationed at the Beersheba complex. When the cyber talented officially leave the army they are funded to start cyber companies with technology developed by themselves or given to them by the state. Much of this is stolen through backdoors of computer systems around the world with America top of the list. Others are sent off to Silicon Valley to start companies or join the major ones and so we have many major positions filled by apparently 'Jewish' but really Sabbatian operatives. Google, YouTube and Facebook are all run by 'Jewish' CEOs while Twitter is all but run by ultra-Zionist hedge-fund shark Paul Singer. At the centre of the Sabbatian global cyber web is the Israeli army's Unit 8200 which specialises in hacking into computer systems of other countries, inserting viruses, gathering information, instigating malfunction, and even taking control of them from a distance. A long list of Sabbatians involved with 9/11, Silicon Valley and Israeli cyber security companies are operatives of Unit 8200. This is not about Israel. It's about the Cult. Israel is planned to be a Smart Grid hub as with China and what is happening at Beersheba is not for the benefit of Jewish people who are treated disgustingly by the Sabbatian elite that control the country. A glance at the Nuremberg Codes will tell you that.

The story is much bigger than 'Covid', important as that is to where we are being taken. Now, though, it's time to really strap in. There's more … much more …

CHAPTER ELEVEN

Who controls the Cult?

Awake, arise or be forever fall'n
John Milton, *Paradise Lost*

I have exposed this far the level of the Cult conspiracy that operates in the world of the seen and within the global secret society and satanic network which operates in the shadows one step back from the seen. The story, however, goes much deeper than that.

The 'Covid' hoax is major part of the Cult agenda, but only part, and to grasp the biggest picture we have to expand our attention beyond the realm of human sight and into the infinity of possibility that we cannot see. It is from here, ultimately, that humanity is being manipulated into a state of total control by the force which dictates the actions of the Cult. How much of reality can we see? Next to damn all is the answer. We may appear to see all there is to see in the 'space' our eyes survey and observe, but little could be further from the truth. The human 'world' is only a tiny band of frequency that the body's visual and perceptual systems can decode into *perception* of a 'world'. According to mainstream science the electromagnetic spectrum is 0.005 percent of what exists in the Universe (Fig 10). The maximum estimate I have seen is 0.5 percent and either way it's miniscule. I say it is far, far, smaller even than 0.005 percent when you compare reality we see with the totality of reality that we don't. Now get this if you are new to such information: Visible light, the only band of frequency that we can see, is a *fraction* of the 0.005

percent (Fig 11 overleaf). Take this further and realise that our universe is one of infinite universes and that universes are only a fragment of overall reality – *infinite* reality. Then compare that with the almost infinitesimal frequency band of visible light or human sight. You see that humans are as near blind as it is possible to be without actually being so. Artist and filmmaker, Sergio Toporek, said:

Figure 10: Humans can perceive such a tiny band of visual reality it's laughable.

Figure 11: We can see a smear of the 0.005 percent electromagnetic spectrum, but we still know it all. Yep, makes sense.

Consider that you can see less than 1% of the electromagnetic spectrum and hear less than 1% of the acoustic spectrum. 90% of the cells in your body carry their own microbial DNA and are not 'you'. The atoms in your body are 99.9999999999999999% empty space and none of them are the ones you were born with ... Human beings have 46 chromosomes, two less than a potato.

The existence of the rainbow depends on the conical photoreceptors in your eyes; to animals without cones, the rainbow does not exist. So you don't just look at a rainbow, you create it. This is pretty amazing, especially considering that all the beautiful colours you see represent less than 1% of the electromagnetic spectrum.

Suddenly the 'world' of humans looks a very different place. Take into account, too, that Planet Earth when compared with the projected size of this single universe is the equivalent of a billionth of a pinhead. Imagine the ratio that would be when compared to infinite reality. To think that Christianity once insisted that Earth and humanity were the centre of everything. This background is vital if we are going to appreciate the nature of 'human' and how we can be manipulated by an unseen force. To human visual reality virtually *everything* is unseen and yet the prevailing perception within the institutions and so much of the public is that if we can't see it, touch it, hear it, taste it and smell it then it cannot exist. Such perception is indoctrinated and encouraged by the Cult and its agents because it isolates believers in the strictly limited, village-idiot, realm of the five senses where perceptions can be firewalled and information controlled. Most of those perpetuating the 'this-world-is-all-there-is' insanity are themselves indoctrinated into believing the same delusion. While major players and influencers know that official reality is laughable most of those in science, academia and medicine really believe the nonsense they peddle and teach succeeding generations. Those who challenge the orthodoxy are dismissed as nutters and freaks to protect the manufactured illusion from exposure. Observe the dynamic of the 'Covid' hoax and you will see how that takes the same form. The inner-circle psychopaths knows it's a gigantic scam, but almost the entirety of those imposing their fascist rules believe that 'Covid' is all that they're told it is.

Stolen identity

Ask people who they are and they will give you their name, place of birth, location, job, family background and life story. Yet that is not who they are – it is what they are *experiencing*. The difference is *absolutely crucial*. The true 'I', the eternal, infinite 'I', is consciousness,

a state of being aware. Forget 'form'. That is a vehicle for a brief experience. Consciousness does not come *from* the brain, but *through* the brain and even that is more symbolic than literal. We are awareness, pure awareness, and this is what withdraws from the body at what we call 'death' to continue our eternal beingness, *isness*, in other realms of reality within the limitlessness of infinity or the Biblical 'many mansions in my father's house'. Labels of a human life, man, woman, transgender, black, white, brown, nationality, circumstances and income are not who we are. They are what we are – awareness – is *experiencing* in a brief connection with a band of frequency we call 'human'. The labels are not the self; they are, to use the title of one of my books, a *Phantom* Self. I am not David Icke born in Leicester, England, on April 29th, 1952. I am the consciousness *having that experience*. The Cult and its non-human masters seek to convince us through the institutions of 'education', science, medicine, media and government that what we are *experiencing* is who we *are*. It's so easy to control and direct perception locked away in the bewildered illusions of the five senses with no expanded radar. Try, by contrast, doing the same with a humanity aware of its true self and its true power to consciously create its reality and experience. How is it possible to do this? We do it all day every day. If you perceive yourself as 'little me' with no power to impact upon your life and the world then your life experience will reflect that. You will hand the power you don't think you have to authority in all its forms which will use it to control your experience. This, in turn, will appear to confirm your perception of 'little me' in a self-fulfilling feedback loop. But that is what 'little me' really is – a *perception*. We are all 'big-me', infinite me, and the Cult has to make us forget that if its will is to prevail. We are therefore manipulated and pressured into self-identifying with human labels and not the consciousness/awareness *experiencing* those human labels.

The phenomenon of identity politics is a Cult-instigated manipulation technique to sub-divide previous labels into even smaller ones. A United States university employs this list of letters to

describe student identity: LGBTTQQFAGPBDSM or lesbian, gay, bisexual, transgender, transsexual, queer, questioning, flexual, asexual, gender-fuck, polyamorous, bondage/discipline, dominance/submission and sadism/masochism. I'm sure other lists are even longer by now as people feel the need to self-identity the 'I' with the minutiae of race and sexual preference. Wokers programmed by the Cult for generations believe this is about 'inclusivity' when it's really the Cult locking them away into smaller and smaller versions of Phantom Self while firewalling them from the influence of their true self, the infinite, eternal 'I'. You may notice that my philosophy which contends that we are all unique points of attention/awareness within the same infinite whole or Oneness is the ultimate non-racism. The very sense of Oneness makes the judgement of people by their body-type, colour or sexuality utterly ridiculous and confirms that racism has no understanding of reality (including anti-white racism). Yet despite my perception of life Cult agents and fast-asleep Wokers label me racist to discredit my information while they are themselves phenomenally racist and sexist. All they see is race and sexuality and they judge people as good or bad, demons or untouchables, by their race and sexuality. All they see is *Phantom Self* and perceive themselves in terms of Phantom Self. They are pawns and puppets of the Cult agenda to focus attention and self-identity in the five senses and play those identities against each other to divide and rule. Columbia University has introduced segregated graduations in another version of social distancing designed to drive people apart and teach them that different racial and cultural groups have nothing in common with each other. The last thing the Cult wants is unity. Again the pump-primers of this will be Cult operatives in the knowledge of what they are doing, but the rest are just the Phantom Self blind leading the Phantom Self blind. We *do* have something in common – we are all *the same consciousness* having different temporary experiences.

What is this 'human'?

Yes, what *is* 'human'? That is what we are supposed to be, right? I mean 'human'? True, but 'human' is the experience not the 'I'. Break it down to basics and 'human' is the way that information is processed. If we are to experience and interact with this band of frequency we call the 'world' we must have a vehicle that operates within that band of frequency. Our consciousness in its prime form cannot do that; it is way beyond the frequency of the human realm. My consciousness or awareness could not tap these keys and pick up the cup in front of me in the same way that radio station A cannot interact with radio station B when they are on different frequencies. The human body is the means through which we have that interaction. I have long described the body as a biological computer which processes information in a way that allows consciousness to experience this reality. The body is a receiver, transmitter and processor of information in a particular way that we call human. We visually perceive only the world of the five senses in a wakened state – that is the limit of the body's visual decoding system. In truth it's not even visual in the way we experience 'visual reality' as I will come to in a moment. We are 'human' because the body processes the information sources of human into a reality and behaviour system that we *perceive* as human. Why does an elephant act like an elephant and not like a human or a duck? The elephant's biological computer is a different information field and processes information according to that program into a visual and behaviour type we call an elephant. The same applies to everything in our reality. These body information fields are perpetuated through procreation (like making a copy of a software program). The Cult wants to break that cycle and intervene technologically to transform the human information field into one that will change what we call humanity. If it can change the human information field it will change the way that field processes information and change humanity both 'physically' and psychologically. Hence the *messenger* (information) RNA 'vaccines' and so much more that is targeting human genetics by changing the body's information – *messaging* – construct through food, drink, radiation, toxicity and other means.

Reality that we experience is nothing like reality as it really is in the same way that the reality people experience in virtual reality games is not the reality they are really living in. The game is only a decoded source of information that appears to be a reality. Our world is also an information construct – a *simulation* (more later). In its base form our reality is a wavefield of information much the same in theme as Wi-Fi. The five senses decode wavefield information into electrical information which they communicate to the brain to decode into holographic (illusory 'physical') information. Different parts of the brain specialise in decoding different senses and the information is fused into a reality that appears to be outside of us but is really inside the brain and the genetic structure in general (Fig 12 overleaf). DNA is a receiver-transmitter of information and a vital part of this decoding process and the body's connection to other realities. Change DNA and you change the way we decode and connect with reality – see 'Covid vaccines'. Think of computers decoding Wi-Fi. You have information encoded in a radiation field and the computer decodes that information into a very different form on the screen. You can't see the Wi-Fi until its information is made manifest on the screen and the information on the screen is inside the computer and not outside. I have just described how we decode the 'human world'. All five senses decode the waveform 'Wi-Fi' field into electrical signals and the brain (computer) constructs reality inside the brain and not outside – 'You don't just look at a rainbow, you create it'. Sound is a simple example. We don't hear sound until the brain decodes it. Waveform sound waves are picked up by the hearing sense and communicated to the brain in an electrical form to be decoded into the sounds that we hear. Everything we hear is inside the brain along with everything we see, feel, smell and taste. Words and language are waveform fields generated by our vocal chords which pass through this process until they are decoded by the brain into words that we hear. Different languages are different frequency fields or sound waves generated by vocal chords. Late British philosopher Alan Watts said:

Figure 12: The brain receives information from the five senses and constructs from that our perceived reality.

[Without the brain] the world is devoid of light, heat, weight, solidity, motion, space, time or any other imaginable feature. All these phenomena are interactions, or transactions, of vibrations with a certain arrangement of neurons.

That's exactly what they are and scientist Robert Lanza describes in his book, *Biocentrism*, how we decode electromagnetic waves and energy into visual and 'physical' experience. He uses the example of a flame emitting photons, electromagnetic energy, each pulsing electrically and magnetically:

… these … invisible electromagnetic waves strike a human retina, and if (and only if) the waves happen to measure between 400 and 700 nano meters in length from crest to crest, then their energy is just right to deliver a stimulus to the 8 million cone-shaped cells in the retina.

Each in turn send an electrical pulse to a neighbour neuron, and on up the line this goes, at 250 mph, until it reaches the … occipital lobe of the brain, in the back of the head. There, a cascading complex of neurons fire from the incoming stimuli, and we subjectively perceive this experience as a yellow brightness occurring in a place we have been conditioned to call the 'external world'.

You hear what you decode

If a tree falls or a building collapses they make no noise unless someone is there to decode the energetic waves generated by the disturbance into what we call sound. Does a falling tree make a noise? Only if you hear it – *decode* it. Everything in our reality is a frequency field of information operating within the overall 'Wi-Fi' field that I call The Field. A vibrational disturbance is generated in The Field by the fields of the falling tree or building. These disturbance waves are what we decode into the sound of them falling. If no one is there to do that then neither will make any noise. Reality is created by the observer – *decoder* – and the *perceptions* of the observer affect the decoding process. For this reason different people – different *perceptions* – will perceive the same reality or situation in a different way. What one may perceive as a nightmare another will see as an opportunity. The question of why the Cult is so focused on controlling human perception now answers itself. All experienced reality is the act of decoding and we don't experience Wi-Fi until it is decoded on the computer screen. The sight and sound of an Internet video is encoded in the Wi-Fi all around us, but we don't see or hear it until the computer decodes that information. Taste, smell and touch are all phenomena of the brain as a result of the same process. We don't taste, smell or feel anything except in the brain and there are pain relief techniques that seek to block the signal from the site of discomfort to the brain because if the brain doesn't decode that signal we don't feel pain. Pain is in the brain and only appears to be at the point of impact thanks to the feedback loop between them. We don't see anything until electrical information from the sight senses is decoded in an area at the back of the brain. If that area is damaged we can go blind when our eyes are perfectly okay. So why do we go blind if we damage an eye? We damage the information processing between the waveform visual information and the visual decoding area of the brain. If information doesn't reach the brain in a form it can decode then we can't see the visual reality that it represents. What's more the brain is decoding only a fraction of the information it receives and the rest is absorbed by the

sub-conscious mind. This explanation is from the science magazine, *Wonderpedia*:

> Every second, 11 million sensations crackle along these [brain] pathways ... The brain is confronted with an alarming array of images, sounds and smells which it rigorously filters down until it is left with a manageable list of around 40. Thus 40 sensations per second make up what we perceive as reality.

The 'world' is not what people are told to believe that is it and the inner circles of the Cult *know that*.

Illusory 'physical' reality

We can only see a smear of 0.005 percent of the Universe which is only one of a vast array of universes – 'mansions' – within infinite reality. Even then the brain decodes only 40 pieces of information ('sensations') from a potential *11 million* that we receive every second. Two points strike you from this immediately: The sheer breathtaking stupidity of believing we know anything so rigidly that there's nothing more to know; and the potential for these processes to be manipulated by a malevolent force to control the reality of the population. One thing I can say for sure with no risk of contradiction is that when you can perceive an almost indescribable fraction of infinite reality there is always more to know as in tidal waves of it. Ancient Greek philosopher Socrates was so right when he said that wisdom is to know how little we know. How obviously true that is when you think that we are experiencing a physical world of solidity that is neither physical nor solid and a world of apartness when everything is connected. Cult-controlled 'science' dismisses the so-called 'paranormal' and all phenomena related to that when the 'para'-normal is perfectly normal and explains the alleged 'great mysteries' which dumbfound scientific minds. There is a reason for this. A 'scientific mind' in terms of the mainstream is a material mind, a five-sense mind imprisoned in see it, touch it, hear it, smell it and taste it. Phenomena and happenings that can't be explained that way leave the 'scientific mind' bewildered and the rule is that if they

can't account for why something is happening then it can't, by definition, be happening. I beg to differ. Telepathy is thought waves passing through The Field (think wave disturbance again) to be decoded by someone able to connect with that wavelength (information). For example: You can pick up the thought waves of a friend at any distance and at the very least that will bring them to mind. A few minutes later the friend calls you. 'My god', you say, 'that's incredible – I was just thinking of you.' Ah, but *they* were thinking of *you* before they made the call and that's what you decoded. Native peoples not entrapped in five-sense reality do this so well it became known as the 'bush telegraph'. Those known as psychics and mediums (genuine ones) are doing the same only across dimensions of reality. 'Mind over matter' comes from the fact that matter and mind are the *same*. The state of one influences the state of the other. Indeed one *and* the other are illusions. They are aspects of the same field. Paranormal phenomena are all explainable so why are they still considered 'mysteries' or not happening? Once you go down this road of understanding you begin to expand awareness beyond the five senses and that's the nightmare for the Cult.

Figure 13: Holograms are not solid, but the best ones appear to be.

Figure 14: How holograms are created by capturing a waveform version of the subject image.

Holographic 'solidity'

Our reality is not solid, it is holographic. We are now well aware of holograms which are widely used today. Two-dimensional information is decoded into a three-dimensional reality that is not solid although can very much appear to be (Fig 13). Holograms are created with a laser divided into two parts. One goes directly onto a holographic photographic print ('reference beam') and the other takes a waveform image of the subject ('working beam') before being directed onto the print where it 'collides' with the other half of the laser (Fig 14). This creates a *waveform* interference pattern which contains the wavefield information of whatever is being photographed (Fig 15 overleaf). The process can be likened to dropping pebbles in a pond. Waves generated by each one spread out across the water to collide with the others and create a wave representation of where the stones fell and at what speed, weight and distance. A waveform interference pattern of a hologram is akin to the waveform information in The Field which the five senses decode into electrical signals to be decoded by the brain into a holographic illusory 'physical' reality. In the same way when a laser (think human attention) is directed at the waveform interference pattern a three-dimensional version of the subject is projected into apparently 'solid' reality (Fig 16). An amazing trait of holograms reveals more 'paranormal mysteries'. Information of the *whole*

hologram is encoded in waveform in every part of the interference pattern by the way they are created. This means that every *part* of a hologram is a smaller version of the whole. Cut the interference wave-pattern into four and you won't get four parts of the image. You get quarter-sized versions of the *whole* image. The body is a hologram and the same applies. Here we have the basis of acupuncture, reflexology and other forms of healing which identify representations of the whole body in all of the parts, hands, feet, ears, everywhere. Skilled palm readers can do what they do because the information of whole body is encoded in the hand. The concept of as above, so below, comes from this.

Figure 15: A waveform interference pattern that holds the information that transforms into a hologram.

Figure 16: Holographic people including 'Elvis' holographically inserted to sing a duet with Celine Dion.

The question will be asked of why, if solidity is illusory, we can't just walk through walls and each other. The resistance is not solid against solid; it is electromagnetic field against electromagnetic field and we decode this into the *experience* of solid against solid. We should also not underestimate the power of belief to dictate reality. What you believe is impossible *will be*. Your belief impacts on your decoding processes and they won't decode what you think is impossible. What we believe we perceive and what we perceive we experience. 'Can't dos' and 'impossibles' are like a firewall in a computer system that won't put on the screen what the firewall blocks. How vital that is to understanding how human experience has been hijacked. I explain in *The Answer*, *Everything You Need To Know But Have Never Been Told* and other books a long list of 'mysteries' and 'paranormal' phenomena that are not mysterious and perfectly normal once you realise what reality is and how it works. 'Ghosts' can be seen to pass through 'solid' walls because the walls are not solid and the ghost is a discarnate entity operating on a frequency so different to that of the wall that it's like two radio stations sharing the same space while never interfering with each other. I have seen ghosts do this myself. The apartness of people and objects is also an illusion. Everything is connected by the Field like all sea life is connected by the sea. It's just that within the limits of our visual reality we only 'see' holographic information and not the field of information that connects everything and from which the holographic world is made manifest. If you can only see holographic 'objects' and not the field that connects them they will appear to you as unconnected to each other in the same way that we see the computer while not seeing the Wi-Fi.

What you don't know *can* hurt you

Okay, we return to those 'two worlds' of human society and the Cult with its global network of interconnecting secret societies and satanic groups which manipulate through governments, corporations, media, religions, etc. The fundamental difference between them is *knowledge*. The idea has been to keep humanity

ignorant of the plan for its total enslavement underpinned by a crucial ignorance of reality – who we are and where we are – and how we interact with it. 'Human' should be the interaction between our expanded eternal consciousness and the five-sense body experience. We are meant to be *in* this world in terms of the five senses but not *of* this world in relation to our greater consciousness and perspective. In that state we experience the small picture of the five senses within the wider context of the big picture of awareness beyond the five senses. Put another way the five senses see the dots and expanded awareness connects them into pictures and patterns that give context to the apparently random and unconnected. Without the context of expanded awareness the five senses see only apartness and randomness with apparently no meaning. The Cult and its other-dimensional controllers seek to intervene in the frequency realm where five-sense reality is supposed to connect with expanded reality and to keep the two apart (more on this in the final chapter). When that happens five-sense mental and emotional processes are no longer influenced by expanded awareness, or the True 'I', and instead are driven by the isolated perceptions of the body's decoding systems. They are in the world *and* of it. Here we have the human plight and why humanity with its potential for infinite awareness can be so easily manipulatable and descend into such extremes of stupidity.

Once the Cult isolates five-sense mind from expanded awareness it can then program the mind with perceptions and beliefs by controlling information that the mind receives through the 'education' system of the formative years and the media perceptual bombardment and censorship of an entire lifetime. Limit perception and a sense of the possible through limiting knowledge by limiting and skewing information while censoring and discrediting that which could set people free. As the title of another of my books says *… And The Truth Shall Set You Free*. For this reason the last thing the Cult wants in circulation is the truth about anything – especially the reality of the eternal 'I' – and that's why it is desperate to control information. The Cult knows that information becomes perception

which becomes behaviour which, collectively, becomes human society. Cult-controlled and funded mainstream 'science' denies the existence of an eternal 'I' and seeks to dismiss and trash all evidence to the contrary. Cult-controlled mainstream religion has a version of 'God' that is little more than a system of control and dictatorship that employs threats of damnation in an afterlife to control perceptions and behaviour in the here and now through fear and guilt. Neither is true and it's the 'neither' that the Cult wishes to suppress. This 'neither' is that everything is an expression, a point of attention, within an infinite state of consciousness which is the real meaning of the term 'God'.

Perceptual obsession with the 'physical body' and five-senses means that 'God' becomes personified as a bearded bloke sitting among the clouds or a raging bully who loves us if we do what 'he' wants and condemns us to the fires of hell if we don't. These are no more than a 'spiritual' fairy tales to control and dictate events and behaviour through fear of this 'God' which has bizarrely made 'God-fearing' in religious circles a state to be desired. I would suggest that fearing *anything* is not to be encouraged and celebrated, but rather deleted. You can see why 'God fearing' is so beneficial to the Cult and its religions when *they* decide what 'God' wants and what 'God' demands (the Cult demands) that everyone do. As the great American comedian Bill Hicks said satirising a Christian zealot: 'I think what God meant to say.' How much of this infinite awareness ('God') that we access is decided by how far we choose to expand our perceptions, self-identity and sense of the possible. The scale of self-identity reflects itself in the scale of awareness that we can connect with and are influenced by – how much knowing and insight we have instead of programmed perception. You cannot expand your awareness into the infinity of possibility when you believe that you are little me Peter the postman or Mary in marketing and nothing more. I'll deal with this in the concluding chapter because it's crucial to how we turnaround current events.

Where the Cult came from

When I realised in the early 1990s there was a Cult network behind global events I asked the obvious question: When did it start? I took it back to ancient Rome and Egypt and on to Babylon and Sumer in Mesopotamia, the 'Land Between Two Rivers', in what we now call Iraq. The two rivers are the Tigris and Euphrates and this region is of immense historical and other importance to the Cult, as is the land called Israel only 550 miles away by air. There is much more going with deep esoteric meaning across this whole region. It's not only about 'wars for oil'. Priceless artefacts from Mesopotamia were stolen or destroyed after the American and British invasion of Iraq in 2003 justified by the lies of Boy Bush and Tony Blair (their Cult masters) about non-existent 'weapons of mass destruction'. Mesopotamia was the location of Sumer (about 5,400BC to 1,750BC), and Babylon (about 2,350BC to 539BC). Sabbatians may have become immensely influential in the Cult in modern times but they are part of a network that goes back into the mists of history. Sumer is said by historians to be the 'cradle of civilisation'. I disagree. I say it was the re-start of what we call human civilisation after cataclysmic events symbolised in part as the 'Great Flood' destroyed the world that existed before. These fantastic upheavals that I have been describing in detail in the books since the early1990s appear in accounts and legends of ancient cultures across the world and they are supported by geological and biological evidence. Stone tablets found in Iraq detailing the Sumer period say the cataclysms were caused by non-human 'gods' they call the Anunnaki. These are described in terms of extraterrestrial visitations in which knowledge supplied by the Anunnaki is said to have been the source of at least one of the world's oldest writing systems and developments in astronomy, mathematics and architecture that were way ahead of their time. I have covered this subject at length in *The Biggest Secret* and *Children of the Matrix* and the same basic 'Anunnaki' story can be found in Zulu accounts in South Africa where the late and very great Zulu high shaman Credo Mutwa told me that the Sumerian Anunnaki were known by Zulus as the Chitauri or 'children of the serpent'. See my six-hour video interview with Credo on this subject entitled *The*

Reptilian Agenda recorded at his then home near Johannesburg in 1999 which you can watch on the Ickonic media platform.

The Cult emerged out of Sumer, Babylon and Egypt (and elsewhere) and established the Roman Empire before expanding with the Romans into northern Europe from where many empires were savagely imposed in the form of Cult-controlled societies all over the world. Mass death and destruction was their calling card. The Cult established its centre of operations in Europe and European Empires were Cult empires which allowed it to expand into a global force. Spanish and Portuguese colonialists headed for Central and South America while the British and French targeted North America. Africa was colonised by Britain, France, Belgium, the Netherlands, Portugal, Spain, Italy, and Germany. Some like Britain and France moved in on the Middle East. The British Empire was by far the biggest for a simple reason. By now Britain was the headquarters of the Cult from which it expanded to form Canada, the United States, Australia and New Zealand. The Sun never set on the British Empire such was the scale of its occupation. London remains a global centre for the Cult along with Rome and the Vatican although others have emerged in Israel and China. It is no accident that the 'virus' is alleged to have come out of China while Italy was chosen as the means to terrify the Western population into compliance with 'Covid' fascism. Nor that Israel has led the world in 'Covid' fascism and mass 'vaccination'.

You would think that I would mention the United States here, but while it has been an important means of imposing the Cult's will it is less significant than would appear and is currently in the process of having what power it does have deleted. The Cult in Europe has mostly loaded the guns for the US to fire. America has been controlled from Europe from the start through Cult operatives in Britain and Europe. The American Revolution was an illusion to make it appear that America was governing itself while very different forces were pulling the strings in the form of Cult families such as the Rothschilds through the Rockefellers and other subordinates. The Rockefellers are extremely close to Bill Gates and

established both scalpel and drug 'medicine' and the World Health Organization. They play a major role in the development and circulation of vaccines through the Rockefeller Foundation on which Bill Gates said his Foundation is based. Why wouldn't this be the case when the Rockefellers and Gates are on the same team? Cult infiltration of human society goes way back into what we call history and has been constantly expanding and centralising power with the goal of establishing a global structure to dictate everything. Look how this has been advanced in great leaps with the 'Covid' hoax.

The non-human dimension

I researched and observed the comings and goings of Cult operatives through the centuries and even thousands of years as they were born, worked to promote the agenda within the secret society and satanic networks, and then died for others to replace them. Clearly there had to be a coordinating force that spanned this entire period while operatives who would not have seen the end goal in their lifetimes came and went advancing the plan over millennia. I went in search of that coordinating force with the usual support from the extraordinary synchronicity of my life which has been an almost daily experience since 1990. I saw common themes in religious texts and ancient cultures about a non-human force manipulating human society from the hidden. Christianity calls this force Satan, the Devil and demons; Islam refers to the Jinn or Djinn; Zulus have their Chitauri (spelt in other ways in different parts of Africa); and the Gnostic people in Egypt in the period around and before 400AD referred to this phenomena as the 'Archons', a word meaning rulers in Greek. Central American cultures speak of the 'Predators' among other names and the same theme is everywhere. I will use 'Archons' as a collective name for all of them. When you see how their nature and behaviour is described all these different sources are clearly talking about the same force. Gnostics described the Archons in terms of 'luminous fire' while Islam relates the Jinn to 'smokeless fire'. Some refer to beings in form that could occasionally be seen, but the most common of common theme is that they operate from

unseen realms which means almost all existence to the visual processes of humans. I had concluded that this was indeed the foundation of human control and that the Cult was operating within the human frequency band on behalf of this hidden force when I came across the writings of Gnostics which supported my conclusions in the most extraordinary way.

A sealed earthen jar was found in 1945 near the town of Nag Hammadi about 75-80 miles north of Luxor on the banks of the River Nile in Egypt. Inside was a treasure trove of manuscripts and texts left by the Gnostic people some 1,600 years earlier. They included 13 leather-bound papyrus codices (manuscripts) and more than 50 texts written in Coptic Egyptian estimated to have been hidden in the jar in the period of 400AD although the source of the information goes back much further. Gnostics oversaw the Great or Royal Library of Alexandria, the fantastic depository of ancient texts detailing advanced knowledge and accounts of human history. The Library was dismantled and destroyed in stages over a long period with the death-blow delivered by the Cult-established Roman Church in the period around 415AD. The Church of Rome was the Church of Babylon relocated as I said earlier. Gnostics were not a race. They were a way of perceiving reality. Whenever they established themselves and their information circulated the terrorists of the Church of Rome would target them for destruction. This happened with the Great Library and with the Gnostic Cathars who were burned to death by the psychopaths after a long period of oppression at the siege of the Castle of Monségur in southern France in 1244. The Church has always been terrified of Gnostic information which demolishes the official Christian narrative although there is much in the Bible that supports the Gnostic view if you read it in another way. To anyone studying the texts of what became known as the Nag Hammadi Library it is clear that great swathes of Christian and Biblical belief has its origin with Gnostics sources going back to Sumer. Gnostic themes have been twisted to manipulate the perceived reality of Bible believers. Biblical texts have been in the open for centuries where they could be changed while Gnostic

documents found at Nag Hammadi were sealed away and untouched for 1,600 years. What you see is what they wrote.

Use your *pneuma* not your *nous*

Gnosticism and Gnostic come from 'gnosis' which means knowledge, or rather *secret* knowledge, in the sense of spiritual awareness – knowledge about reality and life itself. The desperation of the Cult's Church of Rome to destroy the Gnostics can be understood when the knowledge they were circulating was the last thing the Cult wanted the population to know. Sixteen hundred years later the same Cult is working hard to undermine and silence me for the same reason. The dynamic between knowledge and ignorance is a constant. 'Time' appears to move on, but essential themes remain the same. We are told to 'use your nous', a Gnostic word for head/brain/intelligence. They said, however, that spiritual awakening or 'salvation' could only be secured by expanding awareness *beyond* what they called *nous* and into *pneuma* or Infinite Self. Obviously as I read these texts the parallels with what I have been saying since 1990 were fascinating to me. There is a universal truth that spans human history and in that case why wouldn't we be talking the same language 16 centuries apart? When you free yourself from the perception program of the five senses and explore expanded realms of consciousness you are going to connect with the same information no matter what the perceived 'era' within a manufactured timeline of a single and tiny range of manipulated frequency. Humans working with 'smart' technology or knocking rocks together in caves is only a timeline appearing to operate within the human frequency band. Expanded awareness and the knowledge it holds have always been there whether the era be Stone Age or computer age. We can only access that knowledge by opening ourselves to its frequency which the five-sense prison cell is designed to stop us doing. Gates, Fauci, Whitty, Vallance, Zuckerberg, Brin, Page, Wojcicki, Bezos, and all the others behind the 'Covid' hoax clearly have a long wait before their range of frequency can make that connection given that an open heart is

crucial to that as we shall see. Instead of accessing knowledge directly through expanded awareness it is given to Cult operatives by the secret society networks of the Cult where it has been passed on over thousands of years outside the public arena. Expanded realms of consciousness is where great artists, composers and writers find their inspiration and where truth awaits anyone open enough to connect with it. We need to go there fast.

Archon hijack

A fifth of the Nag Hammadi texts describe the existence and manipulation of the Archons led by a 'Chief Archon' they call 'Yaldabaoth', or the 'Demiurge', and this is the Christian 'Devil', 'Satan', 'Lucifer', and his demons. Archons in Biblical symbolism are the 'fallen ones' which are also referred to as fallen angels after the angels expelled from heaven according to the Abrahamic religions of Judaism, Christianity and Islam. These angels are claimed to tempt humans to 'sin' ongoing and you will see how accurate that symbolism is during the rest of the book. The theme of 'original sin' is related to the 'Fall' when Adam and Eve were 'tempted by the serpent' and fell from a state of innocence and 'obedience' (connection) with God into a state of disobedience (disconnection). The Fall is said to have brought sin into the world and corrupted everything including human nature. Yaldabaoth, the 'Lord Archon', is described by Gnostics as a 'counterfeit spirit', 'The Blind One', 'The Blind God', and 'The Foolish One'. The Jewish name for Yaldabaoth in Talmudic writings is Samael which translates as 'Poison of God', or 'Blindness of God'. You see the parallels. Yaldabaoth in Islamic belief is the Muslim Jinn devil known as Shaytan – Shaytan is Satan as the same themes are found all over the world in every religion and culture. The 'Lord God' of the Old Testament is the 'Lord Archon' of Gnostic manuscripts and that's why he's such a bloodthirsty bastard. Satan is known by Christians as 'the Demon of Demons' and Gnostics called Yaldabaoth the 'Archon of Archons'. Both are known as 'The Deceiver'. We are talking about the same 'bloke' for sure and these common themes

using different names, storylines and symbolism tell a common tale of the human plight.

Archons are referred to in Nag Hammadi documents as mind parasites, inverters, guards, gatekeepers, detainers, judges, pitiless ones and deceivers. The 'Covid' hoax alone is a glaring example of all these things. The Biblical 'God' is so different in the Old and New Testaments because they are not describing the same phenomenon. The vindictive, angry, hate-filled, 'God' of the Old Testament, known as Yahweh, is Yaldabaoth who is depicted in Cult-dictated popular culture as the 'Dark Lord', 'Lord of Time', Lord (Darth) Vader and Dormammu, the evil ruler of the 'Dark Dimension' trying to take over the 'Earth Dimension' in the Marvel comic movie, *Dr Strange*. Yaldabaoth is both the Old Testament 'god' and the Biblical 'Satan'. Gnostics referred to Yaldabaoth as the 'Great Architect of the Universe'and the Cult-controlled Freemason network calls their god 'the 'Great Architect of the Universe' (also Grand Architect). The 'Great Architect' Yaldabaoth is symbolised by the Cult as the all-seeing eye at the top of the pyramid on the Great Seal of the United States and the dollar bill. Archon is encoded in *arch*-itect as it is in *arch*-angels and *arch*-bishops. All religions have the theme of a force for good and force for evil in some sort of spiritual war and there is a reason for that – the theme is true. The Cult and its non-human masters are quite happy for this to circulate. They present themselves as the force for good fighting evil when they are really the force of evil (absence of love). The whole foundation of Cult modus operandi is inversion. They promote themselves as a force for good and anyone challenging them in pursuit of peace, love, fairness, truth and justice is condemned as a satanic force for evil. This has been the game plan throughout history whether the Church of Rome inquisitions of non-believers or 'conspiracy theorists' and 'anti-vaxxers' of today. The technique is the same whatever the timeline era.

Yaldabaoth is revolting (true)

Yaldabaoth and the Archons are said to have revolted against God with Yaldabaoth claiming to *be* God – the *All That Is*. The Old Testament 'God' (Yaldabaoth) demanded to be worshipped as such: '*I am* the LORD, and there is none else, there is no God beside me' (Isaiah 45:5). I have quoted in other books a man who said he was the unofficial son of the late Baron Philippe de Rothschild of the Mouton-Rothschild wine producing estates in France who died in 1988 and he told me about the Rothschild 'revolt from God'. The man said he was given the name Phillip Eugene de Rothschild and we shared long correspondence many years ago while he was living under another identity. He said that he was conceived through 'occult incest' which (within the Cult) was 'normal and to be admired'. 'Phillip' told me about his experience attending satanic rituals with rich and famous people whom he names and you can see them and the wider background to Cult Satanism in my other books starting with *The Biggest Secret*. Cult rituals are interactions with Archontic 'gods'. 'Phillip' described Baron Philippe de Rothschild as 'a master Satanist and hater of God' and he used the same term 'revolt from God' associated with Yaldabaoth/Satan/Lucifer/the Devil in describing the Sabbatian Rothschild dynasty. 'I played a key role in my family's revolt from God', he said. That role was to infiltrate in classic Sabbatian style the Christian Church, but eventually he escaped the mind-prison to live another life. The Cult has been targeting religion in a plan to make worship of the Archons the global one-world religion. Infiltration of Satanism into modern 'culture', especially among the young, through music videos, stage shows and other means, is all part of this.

Nag Hammadi texts describe Yaldabaoth and the Archons in their prime form as energy – consciousness – and say they can take form if they choose in the same way that consciousness takes form as a human. Yaldabaoth is called 'formless' and represents a deeply inverted, distorted and chaotic state of consciousness which seeks to attached to humans and turn them into a likeness of itself in an attempt at assimilation. For that to happen it has to manipulate

humans into low frequency mental and emotional states that match its own. Archons can certainly appear in human form and this is the origin of the psychopathic personality. The energetic distortion Gnostics called Yaldabaoth *is* psychopathy. When psychopathic Archons take human form that human will be a psychopath as an expression of Yaldabaoth consciousness. Cult psychopaths are Archons in human form. The principle is the same as that portrayed in the 2009 *Avatar* movie when the American military travelled to a fictional Earth-like moon called Pandora in the Alpha Centauri star system to infiltrate a society of blue people, or Na'vi, by hiding within bodies that looked like the Na'vi. Archons posing as humans have a particular hybrid information field, part human, part Archon, (the ancient 'demigods') which processes information in a way that manifests behaviour to match their psychopathic evil, lack of empathy and compassion, and stops them being influenced by the empathy, compassion and love that a fully-human information field is capable of expressing. Cult bloodlines interbreed, be they royalty or dark suits, for this reason and you have their obsession with incest. Interbreeding with full-blown humans would dilute the Archontic energy field that guarantees psychopathy in its representatives in the human realm.

Gnostic writings say the main non-human forms that Archons take are *serpentine* (what I have called for decades 'reptilian' amid unbounded ridicule from the Archontically-programmed) and what Gnostics describe as 'an unborn baby or foetus with grey skin and dark, unmoving eyes'. This is an excellent representation of the ET 'Greys' of UFO folklore which large numbers of people claim to have seen and been abducted by – Zulu shaman Credo Mutwa among them. I agree with those that believe in extraterrestrial or interdimensional visitations today and for thousands of years past. No wonder with their advanced knowledge and technological capability they were perceived and worshipped as gods for technological and other 'miracles' they appeared to perform. Imagine someone arriving in a culture disconnected from the modern world with a smartphone and computer. They would be

seen as a 'god' capable of 'miracles'. The Renegade Mind, however, wants to know the source of everything and not only the way that source manifests as human or non-human. In the same way that a Renegade Mind seeks the original source material for the 'Covid virus' to see if what is claimed is true. The original source of Archons in form is consciousness – the distorted state of consciousness known to Gnostics as Yaldabaoth.

'Revolt from God' is energetic disconnection

Where I am going next will make a lot of sense of religious texts and ancient legends relating to 'Satan', Lucifer' and the 'gods'. Gnostic descriptions sync perfectly with the themes of my own research over the years in how they describe a consciousness distortion seeking to impose itself on human consciousness. I've referred to the core of infinite awareness in previous books as Infinite Awareness in Awareness of Itself. By that I mean a level of awareness that knows that it is all awareness and is aware of all awareness. From here comes the frequency of love in its true sense and balance which is what love is on one level – the balance of all forces into a single whole called Oneness and Isness. The more we disconnect from this state of love that many call 'God' the constituent parts of that Oneness start to unravel and express themselves as a part and not a whole. They become individualised as intellect, mind, selfishness, hatred, envy, desire for power over others, and such like. This is not a problem in the greater scheme in that 'God', the *All That Is*, can experience all these possibilities through different expressions of itself including humans. What we as expressions of the whole experience the *All That Is* experiences. We are the *All That Is* experiencing itself. As we withdraw from that state of Oneness we disconnect from its influence and things can get very unpleasant and very stupid. Archontic consciousness is at the extreme end of that. It has so disconnected from the influence of Oneness that it has become an inversion of unity and love, an inversion of everything, an inversion of life itself. Evil is appropriately live written backwards. Archontic consciousness is obsessed with death, an inversion of life,

and so its manifestations in Satanism are obsessed with death. They use inverted symbols in their rituals such as the inverted pentagram and cross. Sabbatians as Archontic consciousness incarnate invert Judaism and every other religion and culture they infiltrate. They seek disunity and chaos and they fear unity and harmony as they fear love like garlic to a vampire. As a result the Cult, Archons incarnate, act with such evil, psychopathy and lack of empathy and compassion disconnected as they are from the source of love. How could Bill Gates and the rest of the Archontic psychopaths do what they have to human society in the 'Covid' era with all the death, suffering and destruction involved and have no emotional consequence for the impact on others? Now you know. Why have Zuckerberg, Brin, Page, Wojcicki and company callously censored information warning about the dangers of the 'vaccine' while thousands have been dying and having severe, sometimes life-changing reactions? Now you know. Why have Tedros, Fauci, Whitty, Vallance and their like around the world been using case and death figures they're aware are fraudulent to justify lockdowns and all the deaths and destroyed lives that have come from that? Now you know. Why did Christian Drosten produce and promote a 'testing' protocol that he knew couldn't test for infectious disease which led to a global human catastrophe. Now you know. The Archontic mind doesn't give a shit (Fig 17). I personally think that Gates and major Cult insiders are a form of AI cyborg that the Archons want humans to become.

Figure 17: Artist Neil Hague's version of the 'Covid' hierarchy.

Human batteries

A state of such inversion does have its consequences, however. The level of disconnection from the Source of All means that you withdraw from that source of energetic sustenance and creativity. This means that you have to find your own supply of energetic power and it has – *us*. When the Morpheus character in the first *Matrix* movie held up a battery he spoke a profound truth when he said: 'The Matrix is a computer-generated dream world built to keep us under control in order to change the human being into one of

these.' The statement was true in all respects. We do live in a technologically-generated virtual reality simulation (more very shortly) and we have been manipulated to be an energy source for Archontic consciousness. The Disney-Pixar animated movie *Monsters, Inc*. in 2001 symbolised the dynamic when monsters in their world had no energy source and they would enter the human world to terrify children in their beds, catch the child's scream, terror (low-vibrational frequencies), and take that energy back to power the monster world. The lead character you might remember was a single giant eye and the symbolism of the Cult's all-seeing eye was obvious. Every thought and emotion is broadcast as a frequency unique to that thought and emotion. Feelings of love and joy, empathy and compassion, are high, quick, frequencies while fear, depression, anxiety, suffering and hate are low, slow, dense frequencies. Which kind do you think Archontic consciousness can connect with and absorb? In such a low and dense frequency state there's no way it can connect with the energy of love and joy. Archons can only feed off energy compatible with their own frequency and they and their Cult agents want to delete the human world of love and joy and manipulate the transmission of low vibrational frequencies through low-vibrational human mental and emotional states. *We are their energy source*. Wars are energetic banquets to the Archons – a world war even more so – and think how much low-frequency mental and emotional energy has been generated from the consequences for humanity of the 'Covid' hoax orchestrated by Archons incarnate like Gates.

The ancient practice of human sacrifice 'to the gods', continued in secret today by the Cult, is based on the same principle. 'The gods' are Archontic consciousness in different forms and the sacrifice is induced into a state of intense terror to generate the energy the Archontic frequency can absorb. Incarnate Archons in the ritual drink the blood which contains an adrenaline they crave which floods into the bloodstream when people are terrorised. Most of the sacrifices, ancient and modern, are children and the theme of 'sacrificing young virgins to the gods' is just code for children. They

have a particular pre-puberty energy that Archons want more than anything and the energy of the young in general is their target. The California Department of Education wants students to chant the names of Aztec gods (Archontic gods) once worshipped in human sacrifice rituals in a curriculum designed to encourage them to 'challenge racist, bigoted, discriminatory, imperialist/colonial beliefs', join 'social movements that struggle for social justice', and 'build new possibilities for a post-racist, post-systemic racism society'. It's the usual Woke crap that inverts racism and calls it anti-racism. In this case solidarity with 'indigenous tribes' is being used as an excuse to chant the names of 'gods' to which people were sacrificed (and still are in secret). What an example of Woke's inability to see beyond black and white, us and them, They condemn the colonisation of these tribal cultures by Europeans (quite right), but those cultures sacrificing people including children to their 'gods', and mass murdering untold numbers as the Aztecs did, is just fine. One chant is to the Aztec god Tezcatlipoca who had a man sacrificed to him in the 5th month of the Aztec calendar. His heart was cut out and he was eaten. Oh, that's okay then. Come on children … after three … Other sacrificial 'gods' for the young to chant their allegiance include Quetzalcoatl, Huitzilopochtli and Xipe Totec. The curriculum says that 'chants, affirmations, and energizers can be used to bring the class together, build unity around ethnic studies principles and values, and to reinvigorate the class following a lesson that may be emotionally taxing or even when student engagement may appear to be low'. Well, that's the cover story, anyway. Chanting and mantras are the repetition of a particular frequency generated from the vocal cords and chanting the names of these Archontic 'gods' tunes you into their frequency. That is the last thing you want when it allows for energetic synchronisation, attachment and perceptual influence. Initiates chant the names of their 'Gods' in their rituals for this very reason.

Vampires of the Woke

Paedophilia is another way that Archons absorb the energy of children. Paedophiles possessed by Archontic consciousness are used as the conduit during sexual abuse for discarnate Archons to vampire the energy of the young they desire so much. Stupendous numbers of children disappear every year never to be seen again although you would never know from the media. Imagine how much low-vibrational energy has been generated by children during the 'Covid' hoax when so many have become depressed and psychologically destroyed to the point of killing themselves. Shocking numbers of children are now taken by the state from loving parents to be handed to others. I can tell you from long experience of researching this since 1996 that many end up with paedophiles and assets of the Cult through corrupt and Cult-owned social services which in the reframing era has hired many psychopaths and emotionless automatons to do the job. Children are even stolen to order using spurious reasons to take them by the corrupt and secret (because they're corrupt) 'family courts'. I have written in detail in other books, starting with *The Biggest Secret* in 1997, about the ubiquitous connections between the political, corporate, government, intelligence and military elites (Cult operatives) and Satanism and paedophilia. If you go deep enough both networks have an interlocking leadership. The Woke mentality has been developed by the Cult for many reasons: To promote almost every aspect of its agenda; to hijack the traditional political left and turn it fascist; to divide and rule; and to target agenda pushbackers. But there are other reasons which relate to what I am describing here. How many happy and joyful Wokers do you ever see especially at the extreme end? They are a mental and psychological mess consumed by emotional stress and constantly emotionally cocked for the next explosion of indignation at someone referring to a female as a female. They are walking, talking, batteries as Morpheus might say emitting frequencies which both enslave them in low-vibrational bubbles of perceptual limitation and feed the Archons. Add to this the hatred claimed to be love; fascism claimed to 'anti-fascism', racism claimed to be 'anti-racism';

exclusion claimed to inclusion; and the abuse-filled Internet trolling. You have a purpose-built Archontic energy system with not a wind turbine in sight and all founded on Archontic *inversion*. We have whole generations now manipulated to serve the Archons with their actions and energy. They will be doing so their entire adult lives unless they snap out of their Archon-induced trance. Is it really a surprise that Cult billionaires and corporations put so much money their way? Where is the energy of joy and laughter, including laughing at yourself which is confirmation of your own emotional security? Mark Twain said: 'The human race has one really effective weapon, and that is laughter.' We must use it all the time. Woke has destroyed comedy because it has no humour, no joy, sense of irony, or self-deprecation. Its energy is dense and intense. *Mmmmm*, lunch says the Archontic frequency. Rudolf Steiner (1861-1925) was the Austrian philosopher and famous esoteric thinker who established Waldorf education or Steiner schools to treat children like unique expressions of consciousness and not minds to be programmed with the perceptions determined by authority. I'd been writing about this energy vampiring for decades when I was sent in 2016 a quote by Steiner. He was spot on:

> There are beings in the spiritual realms for whom anxiety and fear emanating from human beings offer welcome food. When humans have no anxiety and fear, then these creatures starve. If fear and anxiety radiates from people and they break out in panic, then these creatures find welcome nutrition and they become more and more powerful. These beings are hostile towards humanity. Everything that feeds on negative feelings, on anxiety, fear and superstition, despair or doubt, are in reality hostile forces in super-sensible worlds, launching cruel attacks on human beings, while they are being fed ... These are exactly the feelings that belong to contemporary culture and materialism; because it estranges people from the spiritual world, it is especially suited to evoke hopelessness and fear of the unknown in people, thereby calling up the above mentioned hostile forces against them.

Pause for a moment from this perspective and reflect on what has happened in the world since the start of 2020. Not only will pennies drop, but billion dollar bills. We see the same theme from Don Juan Matus, a Yaqui Indian shaman in Mexico and the information source for Peruvian-born writer, Carlos Castaneda, who wrote a series of

books from the 1960s to 1990s. Don Juan described the force manipulating human society and his name for the Archons was the predator:

> We have a predator that came from the depths of the cosmos and took over the rule of our lives. Human beings are its prisoners. The predator is our lord and master. It has rendered us docile, helpless. If we want to protest, it suppresses our protest. If we want to act independently, it demands that we don't do so ... indeed we are held prisoner!
>
> They took us over because we are food to them, and they squeeze us mercilessly because we are their sustenance. Just as we rear chickens in coops, the predators rear us in human coops, humaneros. Therefore, their food is always available to them.
>
> Different cultures, different eras, same recurring theme.

The 'ennoia' dilemma

Nag Hammadi Gnostic manuscripts say that Archon consciousness has no 'ennoia'. This is directly translated as 'intentionality', but I'll use the term 'creative imagination'. The *All That Is* in awareness of itself is the source of all creativity – all possibility – and the more disconnected you are from that source the more you are subsequently denied 'creative imagination'. Given that Archon consciousness is almost entirely disconnected it severely lacks creativity and has to rely on far more mechanical processes of thought and exploit the creative potential of those that do have 'ennoia'. You can see cases of this throughout human society. Archon consciousness almost entirely dominates the global banking system and if we study how that system works you will appreciate what I mean. Banks manifest 'money' out of nothing by issuing lines of 'credit' which is 'money' that has never, does not, and will never exist except in theory. It's a confidence trick. If you think 'credit' figures-on-a-screen 'money' is worth anything you accept it as payment. If you don't then the whole system collapses through lack of confidence in the value of that 'money'. Archontic bankers with no 'ennoia' are 'lending' 'money' that doesn't exist to humans that *do* have creativity – those that have the inspired ideas and create businesses and products. Archon banking feeds off human creativity

which it controls through 'money' creation and debt. Humans have the creativity and Archons exploit that for their own benefit and control while having none themselves. Archon Internet platforms like Facebook claim joint copyright of everything that creative users post and while Archontic minds like Zuckerberg may officially head that company it will be human creatives on the staff that provide the creative inspiration. When you have limitless 'money' you can then buy other companies established by creative humans. Witness the acquisition record of Facebook, Google and their like. Survey the Archon-controlled music industry and you see non-creative dark suit executives making their fortune from the human creativity of their artists. The cases are endless. Research the history of people like Gates and Zuckerberg and how their empires were built on exploiting the creativity of others. Archon minds cannot create out of nothing, but they are skilled (because they have to be) in what Gnostic texts call 'countermimicry'. They can imitate, but not innovate. Sabbatians trawl the creativity of others through backdoors they install in computer systems through their cybersecurity systems. Archon-controlled China is globally infamous for stealing intellectual property and I remember how Hong Kong, now part of China, became notorious for making counterfeit copies of the creativity of others – 'countermimicry'. With the now pervasive and all-seeing surveillance systems able to infiltrate any computer you can appreciate the potential for Archons to vampire the creativity of humans. Author John Lamb Lash wrote in his book about the Nag Hammadi texts, *Not In His Image*:

> Although they cannot originate anything, because they lack the divine factor of ennoia (intentionality), Archons can imitate with a vengeance. Their expertise is simulation (HAL, virtual reality). The Demiurge [Yaldabaoth] fashions a heaven world copied from the fractal patterns [of the original] ... His construction is celestial kitsch, like the fake Italianate villa of a Mafia don complete with militant angels to guard every portal.

This brings us to something that I have been speaking about since the turn of the millennium. Our reality is a simulation; a virtual reality that we think is real. No, I'm not kidding.

Human reality? Well, virtually

I had pondered for years about whether our reality is 'real' or some kind of construct. I remembered being immensely affected on a visit as a small child in the late 1950s to the then newly-opened Planetarium on the Marylebone Road in London which is now closed and part of the adjacent Madame Tussauds wax museum. It was in the middle of the day, but when the lights went out there was the night sky projected in the Planetarium's domed ceiling and it appeared to be so real. The experience never left me and I didn't know why until around the turn of the millennium when I became certain that our 'night sky' and entire reality is a projection, a virtual reality, akin to the illusory world portrayed in the *Matrix* movies. I looked at the sky one day in this period and it appeared to me like the domed roof of the Planetarium. The release of the first *Matrix* movie in 1999 also provided a synchronistic and perfect visual representation of where my mind had been going for a long time. I hadn't come across the Gnostic Nag Hammadi texts then. When I did years later the correlation was once again astounding. As I read Gnostic accounts from 1,600 years and more earlier it was clear that they were describing the same simulation phenomenon. They tell how the Yaldabaoth 'Demiurge' and Archons created a 'bad copy' of original reality to rule over all that were captured by its illusions and the body was a prison to trap consciousness in the 'bad copy' fake reality. Read how Gnostics describe the 'bad copy' and update that to current times and they are referring to what we would call today a virtual reality simulation.

Author John Lamb Lash said 'the Demiurge fashions a heaven world copied from the fractal patterns' of the original through expertise in 'HAL' or virtual reality simulation. Fractal patterns are part of the energetic information construct of our reality, a sort of blueprint. If these patterns were copied in computer terms it would indeed give you a copy of a 'natural' reality in a non-natural frequency and digital form. The principle is the same as making a copy of a website. The original website still exists, but now you can change the copy version to make it whatever you like and it can

become very different to the original website. Archons have done this with our reality, a *synthetic* copy of prime reality that still exists beyond the frequency walls of the simulation. Trapped within the illusions of this synthetic Matrix, however, were and are human consciousness and other expressions of prime reality and this is why the Archons via the Cult are seeking to make the human body synthetic and give us synthetic AI minds to complete the job of turning the entire reality synthetic including what we perceive to be the natural world. To quote Kurzweil: 'Nanobots will infuse all the matter around us with information. Rocks, trees, everything will become these intelligent creatures.' Yes, *synthetic* 'creatures' just as 'Covid' and other genetically-manipulating 'vaccines' are designed to make the human body synthetic. From this perspective it is obvious why Archons and their Cult are so desperate to infuse synthetic material into every human with their 'Covid' scam.

Let there be (electromagnetic) light

Yaldabaoth, the force that created the simulation, or Matrix, makes sense of the Gnostic reference to 'The Great Architect' and its use by Cult Freemasonry as the name of its deity. The designer of the Matrix in the movies is called 'The Architect' and that trilogy is jam-packed with symbolism relating to these subjects. I have contended for years that the angry Old Testament God (Yaldabaoth) is the 'God' being symbolically 'quoted' in the opening of Genesis as 'creating the world'. This is not the creation of prime reality – it's the creation of the *simulation*. The Genesis 'God' says: 'Let there be Light: and there was light.' But what is this 'Light'? I have said for decades that the speed of light (186,000 miles per second) is not the fastest speed possible as claimed by mainstream science and is in fact the frequency walls or outer limits of the Matrix. You can't have a fastest or slowest anything within all possibility when everything is possible. The human body is encoded to operate within the speed of light or *within the simulation* and thus we see only the tiny frequency band of visible *light*. Near-death experiencers who perceive reality outside the body during temporary 'death' describe a very different

form of light and this is supported by the Nag Hammadi texts. Prime reality beyond the simulation ('Upper Aeons' to the Gnostics) is described as a realm of incredible beauty, bliss, love and harmony – a realm of 'watery light' that is so powerful 'there are no shadows'. Our false reality of Archon control, which Gnostics call the 'Lower Aeons', is depicted as a realm with a different kind of 'light' and described in terms of chaos, 'Hell', 'the Abyss' and 'Outer Darkness', where trapped souls are tormented and manipulated by demons (relate that to the 'Covid' hoax alone). The watery light theme can be found in near-death accounts and it is not the same as *simulation* 'light' which is electromagnetic or radiation light within the speed of light – the 'Lower Aeons'. Simulation 'light' is the 'luminous fire' associated by Gnostics with the Archons. The Bible refers to Yaldabaoth as 'that old serpent, called the Devil, and Satan, which deceiveth the whole world' (Revelation 12:9). I think that making a simulated copy of prime reality ('countermimicry') and changing it dramatically while all the time manipulating humanity to believe it to be real could probably meet the criteria of deceiving the whole world. Then we come to the Cult god Lucifer – the *Light Bringer*. Lucifer is symbolic of Yaldabaoth, the bringer of radiation light that forms the bad copy simulation within the speed of light. 'He' is symbolised by the lighted torch held by the Statue of Liberty and in the name 'Illuminati'. Sabbatian-Frankism declares that Lucifer is the true god and Lucifer is the real god of Freemasonry honoured as their 'Great or Grand Architect of the Universe' (simulation).

I would emphasise, too, the way Archontic technologically-generated luminous fire of radiation has deluged our environment since I was a kid in the 1950s and changed the nature of The Field with which we constantly interact. Through that interaction technological radiation is changing us. The Smart Grid is designed to operate with immense levels of communication power with 5G expanding across the world and 6G, 7G, in the process of development. Radiation is the simulation and the Archontic manipulation system. Why wouldn't the Archon Cult wish to unleash radiation upon us to an ever-greater extreme to form

Kurzweil's 'cloud'? The plan for a synthetic human is related to the need to cope with levels of radiation beyond even anything we've seen so far. Biological humans would not survive the scale of radiation they have in their script. The Smart Grid is a technological sub-reality within the technological simulation to further disconnect five-sense perception from expanded consciousness. It's a technological prison of the mind.

Infusing the 'spirit of darkness'

A recurring theme in religion and native cultures is the manipulation of human genetics by a non-human force and most famously recorded as the biblical 'sons of god' (the god*s* plural in the original) who interbred with the daughters of men. The Nag Hammadi *Apocryphon of John* tells the same story this way:

> He [Yaldabaoth] sent his angels [Archons/demons] to the daughters of men, that they might take some of them for themselves and raise offspring for their enjoyment. And at first they did not succeed. When they had no success, they gathered together again and they made a plan together ... And the angels changed themselves in their likeness into the likeness of their mates, filling them with the spirit of darkness, which they had mixed for them, and with evil ... And they took women and begot children out of the darkness according to the likeness of their spirit.

Possession when a discarnate entity takes over a human body is an age-old theme and continues today. It's very real and I've seen it. Satanic and secret society rituals can create an energetic environment in which entities can attach to initiates and I've heard many stories of how people have changed their personality after being initiated even into lower levels of the Freemasons. I have been inside three Freemasonic temples, one at a public open day and two by just walking in when there was no one around to stop me. They were in Ryde, the town where I live, Birmingham, England, when I was with a group, and Boston, Massachusetts. They all felt the same energetically – dark, dense, low-vibrational and sinister. Demonic attachment can happen while the initiate has no idea what is going on. To them it's just a ritual to get in the Masons and do a bit of good

business. In the far more extreme rituals of Satanism human possession is even more powerful and they are designed to make possession possible. The hierarchy of the Cult is dictated by the power and perceived status of the possessing Archon. In this way the Archon hierarchy becomes the Cult hierarchy. Once the entity has attached it can influence perception and behaviour and if it attaches to the extreme then so much of its energy (information) infuses into the body information field that the hologram starts to reflect the nature of the possessing entity. This is the *Exorcist* movie type of possession when facial features change and it's known as shapeshifting. Islam's Jinn are said to be invisible tricksters who change shape, 'whisper', confuse and take human form. These are all traits of the Archons and other versions of the same phenomenon. Extreme possession could certainty infuse the 'spirit of darkness' into a partner during sex as the Nag Hammadi texts appear to describe. Such an infusion can change genetics which is also energetic information. Human genetics is information and the 'spirit of darkness' is information. Mix one with the other and change must happen. Islam has the concept of a 'Jinn baby' through possession of the mother and by Jinn taking human form. There are many ways that human genetics can be changed and remember that Archons have been aware all along of advanced techniques to do this. What is being done in human society today – and far more – was known about by Archons at the time of the 'fallen ones' and their other versions described in religions and cultures.

Archons and their human-world Cult are obsessed with genetics as we see today and they know this dictates how information is processed into perceived reality during a human life. They needed to produce a human form that would decode the simulation and this is symbolically known as 'Adam and Eve' who left the 'garden' (prime reality) and 'fell' into Matrix reality. The simulation is not a 'physical' construct (there is no 'physical'); it is a source of information. Think Wi-Fi again. The simulation is an energetic field encoded with information and body-brain systems are designed to decode that information encoded in wave or frequency form which

is transmitted to the brain as electrical signals. These are decoded by the brain to construct our sense of reality – an illusory 'physical' world that only exists in the brain or the mind. Virtual reality games mimic this process using the same sensory decoding system. Information is fed to the senses to decode a virtual reality that can appear so real, but isn't (Figs 18 and 19). Some scientists believe – and I agree with them – that what we perceive as 'physical' reality only exists when we are looking or observing. The act of perception or focus triggers the decoding systems which turn waveform information into holographic reality. When we are not observing something our reality reverts from a holographic state to a waveform state. This relates to the same principle as a falling tree not making a noise unless someone is there to hear it or decode it. The concept makes sense from the simulation perspective. A computer is not decoding all the information in a Wi-Fi field all the time and only decodes or brings into reality on the screen that part of Wi-Fi that it's decoding – focusing upon – at that moment.

Figure 18: Virtual reality technology 'hacks' into the body's five-sense decoding system.

Figure 19: The result can be experienced as very 'real'.

Interestingly, Professor Donald Hoffman at the Department of Cognitive Sciences at the University of California, Irvine, says that our experienced reality is like a computer interface that shows us only the level with which we interact while hiding all that exists beyond it: 'Evolution shaped us with a user interface that hides the truth. Nothing that we see is the truth – the very language of space and time and objects is the wrong language to describe reality.' He is correct in what he says on so many levels. Space and time are not a universal reality. They are a phenomenon of decoded *simulation* reality as part of the process of enslaving our sense of reality. Near-death experiencers report again and again how space and time did not exist as we perceive them once they were free of the body – body decoding systems. You can appreciate from this why Archons and their Cult are so desperate to entrap human attention in the five senses where we are in the Matrix and of the Matrix. Opening your mind to expanded states of awareness takes you beyond the information confines of the simulation and you become aware of knowledge and insights denied to you before. This is what we call 'awakening' – *awakening from the Matrix* – and in the final chapter I will relate this to current events.

Where are the 'aliens'?

A simulation would explain the so-called 'Fermi Paradox' named after Italian physicist Enrico Fermi (1901-1954) who created the first nuclear reactor. He considered the question of why there is such a lack of extraterrestrial activity when there are so many stars and planets in an apparently vast universe; but what if the night sky that we see, or think we do, is a simulated projection as I say? If you control the simulation and your aim is to hold humanity fast in essential ignorance would you want other forms of life including advanced life coming and going sharing information with humanity? Or would you want them to believe they were isolated and apparently alone? Themes of human isolation and apartness are common whether they be the perception of a lifeless universe or the fascist isolation laws of the 'Covid' era. Paradoxically the very

existence of a simulation means that we are not alone when some force had to construct it. My view is that experiences that people have reported all over the world for centuries with Reptilians and Grey entities are Archon phenomena as Nag Hammadi texts describe; and that benevolent 'alien' interactions are non-human groups that come in and out of the simulation by overcoming Archon attempts to keep them out. It should be highlighted, too, that Reptilians and Greys are obsessed with *genetics* and *technology* as related by cultural accounts and those who say they have been abducted by them. Technology is their way of overcoming some of the limitations in their creative potential and our technology-driven and controlled human society of today is *arch*etypical Archon-Reptilian-Grey modus operandi. Technocracy is really *Archon*tocracy. The Universe does not have to be as big as it appears with a simulation. There is no space or distance only information decoded into holographic reality. What we call 'space' is only the absence of holographic 'objects' and that 'space' is The Field of energetic information which connects everything into a single whole. The same applies with the artificially-generated information field of the simulation. The Universe is not big or small as a physical reality. It is decoded information, that's all, and its perceived size is decided by the way the simulation is encoded to make it appear. The entire night sky as we perceive it only exists in our brain and so where are those 'millions of light years'? The 'stars' on the ceiling of the Planetarium looked a vast distance away.

There's another point to mention about 'aliens'. I have been highlighting since the 1990s the plan to stage a fake 'alien invasion' to justify the centralisation of global power and a world military. Nazi scientist Werner von Braun, who was taken to America by Operation Paperclip after World War Two to help found NASA, told his American assistant Dr Carol Rosin about the Cult agenda when he knew he was dying in 1977. Rosin said that he told her about a sequence that would lead to total human control by a one-world government. This included threats from terrorism, rogue nations, meteors and asteroids before finally an 'alien invasion'. All of these

things, von Braun said, would be bogus and what I would refer to as a No-Problem-Reaction-Solution. Keep this in mind when 'the aliens are coming' is the new mantra. The aliens are not coming – they are *already here* and they have infiltrated human society while looking human. French-Canadian investigative journalist Serge Monast said in 1994 that he had uncovered a NASA/military operation called Project Blue Beam which fits with what Werner von Braun predicted. Monast died of a 'heart attack' in 1996 the day after he was arrested and spent a night in prison. He was 51. He said Blue Beam was a plan to stage an alien invasion that would include religious figures beamed holographically into the sky as part of a global manipulation to usher in a 'new age' of worshipping what I would say is the Cult 'god' Yaldabaoth in a one-world religion. Fake holographic asteroids are also said to be part of the plan which again syncs with von Braun. How could you stage an illusory threat from asteroids unless they were holographic inserts? This is pretty straightforward given the advanced technology outside the public arena and the fact that our 'physical' reality is holographic anyway. Information fields would be projected and we would decode them into the illusion of a 'physical' asteroid. If they can sell a global 'pandemic' with a 'virus' that doesn't exist what will humans not believe if government and media tell them?

All this is particularly relevant as I write with the Pentagon planning to release in June, 2021, information about 'UFO sightings'. I have been following the UFO story since the early 1990s and the common theme throughout has been government and military denials and cover up. More recently, however, the Pentagon has suddenly become more talkative and apparently open with Air Force pilot radar images released of unexplained craft moving and changing direction at speeds well beyond anything believed possible with human technology. Then, in March, 2021, former Director of National Intelligence John Ratcliffe said a Pentagon report months later in June would reveal a great deal of information about UFO sightings unknown to the public. He said the report would have 'massive implications'. The order to do this was included bizarrely

in a $2.3 trillion 'coronavirus' relief and government funding bill passed by the Trump administration at the end of 2020. I would add some serious notes of caution here. I have been pointing out since the 1990s that the US military and intelligence networks have long had craft – 'flying saucers' or anti-gravity craft – which any observer would take to be extraterrestrial in origin. Keeping this knowledge from the public allows craft flown by *humans* to be perceived as alien visitations. I am not saying that 'aliens' do not exist. I would be the last one to say that, but we have to be streetwise here. President Ronald Reagan told the UN General Assembly in 1987: 'I occasionally think how quickly our differences worldwide would vanish if we were facing an alien threat from outside this world.' That's the idea. Unite against a common 'enemy' with a common purpose behind your 'saviour force' (the Cult) as this age-old technique of mass manipulation goes global.

Science moves this way ...

I could find only one other person who was discussing the simulation hypothesis publicly when I concluded it was real. This was Nick Bostrom, a Swedish-born philosopher at the University of Oxford, who has explored for many years the possibility that human reality is a computer simulation although his version and mine are not the same. Today the simulation and holographic reality hypothesis have increasingly entered the scientific mainstream. Well, the more open-minded mainstream, that is. Here are a few of the ever-gathering examples. American nuclear physicist Silas Beane led a team of physicists at the University of Bonn in Germany pursuing the question of whether we live in a simulation. They concluded that we probably do and it was likely based on a lattice of cubes. They found that cosmic rays align with that specific pattern. The team highlighted the Greisen–Zatsepin–Kuzmin (GZK) limit which refers to cosmic ray particle interaction with cosmic background radiation that creates an apparent boundary for cosmic ray particles. They say in a paper entitled 'Constraints on the Universe as a Numerical Simulation' that this 'pattern of constraint' is exactly what you

would find with a computer simulation. They also made the point that a simulation would create its own 'laws of physics' that would limit possibility. I've been making the same point for decades that the *perceived* laws of physics relate only to this reality, or what I would later call the simulation. When designers write codes to create computer and virtual reality games they are the equivalent of the laws of physics for that game. Players interact within the limitations laid out by the coding. In the same way those who wrote the codes for the simulation decided the laws of physics that would apply. These can be overridden by expanded states of consciousness, but not by those enslaved in only five-sense awareness where simulation codes rule. Overriding the codes is what people call 'miracles'. They are not. They are bypassing the encoded limits of the simulation. A population caught in simulation perception would have no idea that this was their plight. As the Bonn paper said: 'Like a prisoner in a pitch-black cell we would not be able to see the "walls" of our prison,' That's true if people remain mesmerised by the five senses. Open to expanded awareness and those walls become very clear. The main one is the speed of light.

American theoretical physicist James Gates is another who has explored the simulation question and found considerable evidence to support the idea. Gates was Professor of Physics at the University of Maryland, Director of The Center for String and Particle Theory, and on Barack Obama's Council of Advisors on Science and Technology. He and his team found *computer codes* of digital data embedded in the fabric of our reality. They relate to on-off electrical charges of 1 and 0 in the binary system used by computers. 'We have no idea what they are doing there', Gates said. They found within the energetic fabric mathematical sequences known as error-correcting codes or block codes that 'reboot' data to its original state or 'default settings' when something knocks it out of sync. Gates was asked if he had found a set of equations embedded in our reality indistinguishable from those that drive search engines and browsers and he said: 'That is correct.' Rich Terrile, director of the Centre for Evolutionary Computation and Automated Design at NASA's Jet

Propulsion Laboratory, has said publicly that he believes the Universe is a digital hologram that must have been created by a form of intelligence. I agree with that in every way. Waveform information is delivered electrically by the senses to the brain which constructs a *digital* holographic reality that we call the 'world'. This digital level of reality can be read by the esoteric art of numerology. Digital holograms are at the cutting edge of holographics today. We have digital technology everywhere designed to access and manipulate our digital level of perceived reality. Synthetic mRNA in 'Covid vaccines' has a digital component to manipulate the body's digital 'operating system'.

Reality is numbers

How many know that our reality can be broken down to numbers and codes that are the same as computer games? Max Tegmark, a physicist at the Massachusetts Institute of Technology (MIT), is the author of *Our Mathematical Universe* in which he lays out how reality can be entirely described by numbers and maths in the way that a video game is encoded with the 'physics' of computer games. Our world and computer virtual reality are essentially the same. Tegmark imagines the perceptions of characters in an advanced computer game when the graphics are so good they don't know they are in a game. They think they can bump into real objects (electromagnetic resistance in our reality), fall in love and feel emotions like excitement. When they began to study the apparently 'physical world' of the video game they would realise that everything was made of pixels (which have been found in our energetic reality as must be the case when on one level our world is digital). What computer game characters thought was physical 'stuff', Tegmark said, could actually be broken down into numbers:

> And we're exactly in this situation in our world. We look around and it doesn't seem that mathematical at all, but everything we see is made out of elementary particles like quarks and electrons. And what properties does an electron have? Does it have a smell or a colour or a texture? No! ... We physicists have come up with geeky names for [Electron] properties, like

electric charge, or spin, or lepton number, but the electron doesn't care what we call it, the properties are just numbers.

This is the illusory reality Gnostics were describing. This is the simulation. The A, C, G, and T codes of DNA have a binary value – A and C = 0 while G and T = 1. This has to be when the simulation is digital and the body must be digital to interact with it. Recurring mathematical sequences are encoded throughout reality and the body. They include the Fibonacci sequence in which the two previous numbers are added to get the next one, as in ... 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, etc. The sequence is encoded in the human face and body, proportions of animals, DNA, seed heads, pine cones, trees, shells, spiral galaxies, hurricanes and the number of petals in a flower. The list goes on and on. There are fractal patterns – a 'never-ending pattern that is infinitely complex and self-similar across all scales in the as above, so below, principle of holograms. These and other famous recurring geometrical and mathematical sequences such as Phi, Pi, Golden Mean, Golden Ratio and Golden Section are *computer codes* of the simulation. I had to laugh and give my head a shake the day I finished this book and it went into the production stage. I was sent an article in *Scientific American* published in April, 2021, with the headline 'Confirmed! We Live in a Simulation'. Two decades after I first said our reality is a simulation and the speed of light is it's outer limit the article suggested that we do live in a simulation and that the speed of light is its outer limit. I left school at 15 and never passed a major exam in my life while the writer was up to his eyes in qualifications. As I will explain in the final chapter *knowing* is far better than thinking and they come from very different sources. The article rightly connected the speed of light to the processing speed of the 'Matrix' and said what has been in my books all this time … 'If we are in a simulation, as it appears, then space is an abstract property written in code. It is not real'. No it's not and if we live in a simulation something created it and it wasn't *us*. 'That David Icke says we are manipulated by aliens' – he's crackers.'

Wow ...

The reality that humanity thinks is so real is an illusion. Politicians, governments, scientists, doctors, academics, law enforcement, media, school and university curriculums, on and on, are all founded on a world that *does not exist* except as a simulated prison cell. Is it such a stretch to accept that 'Covid' doesn't exist when our entire 'physical' reality doesn't exist? Revealed here is the knowledge kept under raps in the Cult networks of compartmentalised secrecy to control humanity's sense of reality by inducing the population to believe in a reality that's not real. If it wasn't so tragic in its experiential consequences the whole thing would be hysterically funny. None of this is new to Renegade Minds. Ancient Greek philosopher Plato (about 428 to about 347BC) was a major influence on Gnostic belief and he described the human plight thousands of years ago with his Allegory of the Cave. He told the symbolic story of prisoners living in a cave who had never been outside. They were chained and could only see one wall of the cave while behind them was a fire that they could not see. Figures walked past the fire casting shadows on the prisoners' wall and those moving shadows became their sense of reality. Some prisoners began to study the shadows and were considered experts on them (today's academics and scientists), but what they studied was only an illusion (today's academics and scientists). A prisoner escaped from the cave and saw reality as it really is. When he returned to report this revelation they didn't believe him, called him mad and threatened to kill him if he tried to set them free. Plato's tale is not only a brilliant analogy of the human plight and our illusory reality. It describes, too, the dynamics of the 'Covid' hoax. I have only skimmed the surface of these subjects here. The aim of this book is to crisply connect all essential dots to put what is happening today into its true context. All subject areas and their connections in this chapter are covered in great evidential detail in *Everything You Need To Know, But Have Never Been Told* and *The Answer*.

They say that bewildered people 'can't see the forest for the trees'. Humanity, however, can't see the forest for the *twigs*. The five senses

see only twigs while Renegade Minds can see the forest and it's the forest where the answers lie with the connections that reveals. Breaking free of perceptual programming so the forest can be seen is the way we turn all this around. Not breaking free is how humanity got into this mess. The situation may seem hopeless, but I promise you it's not. We are a perceptual heartbeat from paradise if only we knew.

CHAPTER TWELVE

Escaping Wetiko

Life is simply a vacation from the infinite
Dean Cavanagh

Renegade Minds weave the web of life and events and see common themes in the apparently random. They are always there if you look for them and their pursuit is aided by incredible synchronicity that comes when your mind is open rather than mesmerised by what it thinks it can see.

Infinite awareness is infinite possibility and the more of infinite possibility that we access the more becomes infinitely possible. That may be stating the apparently obvious, but it is a devastatingly-powerful fact that can set us free. We are a point of attention within an infinity of consciousness. The question is how much of that infinity do we choose to access? How much knowledge, insight, awareness, wisdom, do we want to connect with and explore? If your focus is only in the five senses you will be influenced by a fraction of infinite awareness. I mean a range so tiny that it gives new meaning to infinitesimal. Limitation of self-identity and a sense of the possible limit accordingly your range of consciousness. We are what we think we are. Life is what we think it is. The dream is the dreamer and the dreamer is the dream. Buddhist philosophy puts it this way: 'As a thing is viewed, so it appears.' Most humans live in the realm of touch, taste, see, hear, and smell and that's the limit of their sense of the possible and sense of self. Many will follow a religion and speak of a God in his heaven, but their lives are still

dominated by the five senses in their perceptions and actions. The five senses become the arbiter of everything. When that happens all except a smear of infinity is sealed away from influence by the rigid, unyielding, reality bubbles that are the five-sense human or Phantom Self. Archon Cult methodology is to isolate consciousness within five-sense reality – the simulation – and then program that consciousness with a sense of self and the world through a deluge of life-long information designed to instil the desired perception that allows global control. Efforts to do this have increased dramatically with identity politics as identity bubbles are squeezed into the minutiae of five-sense detail which disconnect people even more profoundly from the infinite 'I'.

Five-sense focus and self-identity are like a firewall that limits access to the infinite realms. You only perceive one radio or television station and no other. We'll take that literally for a moment. Imagine a vast array of stations giving different information and angles on reality, but you only ever listen to one. Here we have the human plight in which the population is overwhelmingly confined to CultFM. This relates only to the frequency range of CultFM and limits perception and insight to that band – limits *possibility* to that band. It means you are connecting with an almost imperceptibly minuscule range of possibility and creative potential within the infinite Field. It's a world where everything seems apart from everything else and where synchronicity is rare. Synchronicity is defined in the dictionary as 'the happening by chance of two or more related or similar events at the same time'. Use of 'by chance' betrays a complete misunderstanding of reality. Synchronicity is not 'by chance'. As people open their minds, or 'awaken' to use the term, they notice more and more coincidences in their lives, bits of 'luck', apparently miraculous happenings that put them in the right place at the right time with the right people. Days become peppered with 'fancy meeting you here' and 'what are the chances of that?' My entire life has been lived like this and ever more so since my own colossal awakening in 1990 and 91 which transformed my sense of reality. Synchronicity is not 'by chance'; it is by accessing expanded

realms of possibility which allow expanded potential for manifestation. People broadcasting the same vibe from the same openness of mind tend to be drawn 'by chance' to each other through what I call frequency magnetism and it's not only people. In the last more than 30 years incredible synchronicity has also led me through the Cult maze to information in so many forms and to crucial personal experiences. These 'coincidences' have allowed me to put the puzzle pieces together across an enormous array of subjects and situations. Those who have breached the bubble of five-sense reality will know exactly what I mean and this escape from the perceptual prison cell is open to everyone whenever they make that choice. This may appear super-human when compared with the limitations of 'human', but it's really our natural state. 'Human' as currently experienced is consciousness in an unnatural state of induced separation from the infinity of the whole. I'll come to how this transformation into unity can be made when I have described in more detail the force that holds humanity in servitude by denying this access to infinite self.

The Wetiko factor

I have been talking and writing for decades about the way five-sense mind is systematically barricaded from expanded awareness. I have used the analogy of a computer (five-sense mind) and someone at the keyboard (expanded awareness). Interaction between the computer and the operator is symbolic of the interaction between five-sense mind and expanded awareness. The computer directly experiences the Internet and the operator experiences the Internet via the computer which is how it's supposed to be – the two working as one. Archons seek to control that point where the operator connects with the computer to stop that interaction (Fig 20). Now the operator is banging the keyboard and clicking the mouse, but the computer is not responding and this happens when the computer is taken over – *possessed* – by an appropriately-named computer 'virus'. The operator has lost all influence over the computer which goes its own way making decisions under the control of the 'virus'. I have

just described the dynamic through which the force known to Gnostics as Yaldabaoth and Archons disconnects five-sense mind from expanded awareness to imprison humanity in perceptual servitude.

Figure 20: The mind 'virus' I have been writing about for decades seeks to isolate five-sense mind (the computer) from the true 'I'. (Image by Neil Hague).

About a year ago I came across a Native American concept of Wetiko which describes precisely the same phenomenon. Wetiko is the spelling used by the Cree and there are other versions including wintiko and windigo used by other tribal groups. They spell the name with lower case, but I see Wetiko as a proper noun as with Archons and prefer a capital. I first saw an article about Wetiko by writer and researcher Paul Levy which so synced with what I had been writing about the computer/operator disconnection and later the Archons. I then read his book, the fascinating *Dispelling Wetiko, Breaking the Spell of Evil*. The parallels between what I had concluded long before and the Native American concept of Wetiko were so clear and obvious that it was almost funny. For Wetiko see the Gnostic Archons for sure and the Jinn, the Predators, and every other name for a force of evil, inversion and chaos. Wetiko is the Native American name for the force that divides the computer from

the operator (Fig 21). Indigenous author Jack D. Forbes, a founder of the Native American movement in the 1960s, wrote another book about Wetiko entitled *Columbus And Other Cannibals – The Wetiko Disease of Exploitation, Imperialism, and Terrorism* which I also read. Forbes says that Wetiko refers to an evil person or spirit 'who terrorizes other creatures by means of terrible acts, including cannibalism'. Zulu shaman Credo Mutwa told me that African accounts tell how cannibalism was brought into the world by the Chitauri 'gods' – another manifestation of Wetiko. The distinction between 'evil person or spirit' relates to Archons/Wetiko possessing a human or acting as pure consciousness. Wetiko is said to be a sickness of the soul or spirit and a state of being that takes but gives nothing back – the Cult and its operatives perfectly described. Black Hawk, a Native American war leader defending their lands from confiscation, said European invaders had 'poisoned hearts' – Wetiko hearts – and that this would spread to native societies. Mention of the heart is very significant as we shall shortly see. Forbes writes: 'Tragically, the history of the world for the past 2,000 years is, in great part, the story of the epidemiology of the wetiko disease.' Yes, and much longer. Forbes is correct when he says: 'The wetikos destroyed Egypt and Babylon and Athens and Rome and Tenochtitlan [capital of the Aztec empire] and perhaps now they will destroy the entire earth.' Evil, he said, is the number one export of a Wetiko culture – see its globalisation with 'Covid'. Constant war, mass murder, suffering of all kinds, child abuse, Satanism, torture and human sacrifice are all expressions of Wetiko and the Wetiko possessed. The world is Wetiko made manifest, *but it doesn't have to be*. There is a way out of this even now.

Figure 21: The mind 'virus' is known to Native Americans as 'Wetiko'. (Image by Neil Hague).

Cult of Wetiko

Wetiko is the Yaldabaoth frequency distortion that seeks to attach to human consciousness and absorb it into its own. Once this connection is made Wetiko can drive the perceptions of the target which they believe to be coming from their own mind. All the horrors of history and today from mass killers to Satanists, paedophiles like Jeffrey Epstein and other psychopaths, are the embodiment of Wetiko and express its state of being in all its grotesqueness. The Cult is Wetiko incarnate, Yaldabaoth incarnate, and it seeks to facilitate Wetiko assimilation of humanity in totality into its distortion by manipulating the population into low frequency states that match its own. Paul Levy writes: 'Holographically enforced within the psyche of every human being the wetiko virus pervades and underlies the entire field of consciousness, and can therefore potentially manifest through any one of us at any moment if we are not mindful.' The 'Covid' hoax has achieved this with many people, but others have not fallen into Wetiko's frequency lair. Players in the 'Covid' human catastrophe including Gates, Schwab, Tedros, Fauci, Whitty, Vallance, Johnson, Hancock, Ferguson, Drosten, and all the rest, including the psychopath psychologists, are expressions of Wetiko. This is why

they have no compassion or empathy and no emotional consequence for what they do that would make them stop doing it. Observe all the people who support the psychopaths in authority against the Pushbackers despite the damaging impact the psychopaths have on their own lives and their family's lives. You are again looking at Wetiko possession which prevents them seeing through the lies to the obvious scam going on. *Why can't they see it?* Wetiko won't let them see it. The perceptual divide that has now become a chasm is between the Wetikoed and the non-Wetikoed.

Paul Levy describes Wetiko in the same way that I have long described the Archontic force. They are the same distorted consciousness operating across dimensions of reality: '… the subtle body of wetiko is not located in the third dimension of space and time, literally existing in another dimension … it is able to affect ordinary lives by mysteriously interpenetrating into our three-dimensional world.' Wetiko does this through its incarnate representatives in the Cult and by weaving itself into The Field which on our level of reality is the electromagnetic information field of the simulation or Matrix. More than that, the simulation *is* Wetiko / Yaldabaoth. Caleb Scharf, Director of Astrobiology at Columbia University, has speculated that 'alien life' could be so advanced that it has transcribed itself into the quantum realm to become what we call physics. He said intelligence indistinguishable from the fabric of the Universe would solve many of its greatest mysteries:

> Perhaps hyper-advanced life isn't just external. Perhaps it's already all around. It is embedded in what we perceive to be physics itself, from the root behaviour of particles and fields to the phenomena of complexity and emergence ... In other words, life might not just be in the equations. It might BE the equations [My emphasis].

Scharf said it is possible that 'we don't recognise advanced life because it forms an integral and unsuspicious part of what we've considered to be the natural world'. I agree. Wetiko/Yaldabaoth *is* the simulation. We are literally in the body of the beast. But that doesn't mean it has to control us. We all have the power to overcome Wetiko

influence and the Cult knows that. I doubt it sleeps too well because it knows that.

Which Field?

This, I suggest, is how it all works. There are two Fields. One is the fierce electromagnetic light of the Matrix within the speed of light; the other is the 'watery light' of The Field beyond the walls of the Matrix that connects with the Great Infinity. Five-sense mind and the decoding systems of the body attach us to the Field of Matrix light. They have to or we could not experience this reality. Five-sense mind sees only the Matrix Field of information while our expanded consciousness is part of the Infinity Field. When we open our minds, and most importantly our hearts, to the Infinity Field we have a mission control which gives us an expanded perspective, a road map, to understand the nature of the five-sense world. If we are isolated only in five-sense mind there is no mission control. We're on our own trying to understand a world that's constantly feeding us information to ensure we do not understand. People in this state can feel 'lost' and bewildered with no direction or radar. You can see ever more clearly those who are influenced by the Fields of Big Infinity or little five-sense mind simply by their views and behaviour with regard to the 'Covid' hoax. We have had this division throughout known human history with the mass of the people on one side and individuals who could see and intuit beyond the walls of the simulation – Plato's prisoner who broke out of the cave and saw reality for what it is. Such people have always been targeted by Wetiko/Archon-possessed authority, burned at the stake or demonised as mad, bad and dangerous. The Cult today and its global network of 'anti-hate', 'anti-fascist' Woke groups are all expressions of Wetiko attacking those exposing the conspiracy, 'Covid' lies and the 'vaccine' agenda.

Woke as a whole is Wetiko which explains its black and white mentality and how at one it is with the Wetiko-possessed Cult. Paul Levy said: 'To be in this paradigm is to still be under the thrall of a two-valued logic – where things are either true or false – of a

wetikoized mind.' Wetiko consciousness is in a permanent rage, therefore so is Woke, and then there is Woke inversion and contradiction. 'Anti-fascists' act like fascists because fascists *and* 'anti-fascists' are both Wetiko at work. Political parties act the same while claiming to be different for the same reason. Secret society and satanic rituals are attaching initiates to Wetiko and the cold, ruthless, psychopathic mentality that secures the positions of power all over the world is Wetiko. Reframing 'training programmes' have the same cumulative effect of attaching Wetiko and we have their graduates described as automatons and robots with a cold, psychopathic, uncaring demeanour. They are all traits of Wetiko possession and look how many times they have been described in this book and elsewhere with regard to personnel behind 'Covid' including the police and medical profession. Climbing the greasy pole in any profession in a Wetiko society requires traits of Wetiko to get there and that is particularly true of politics which is not about fair competition and pre-eminence of ideas. It is founded on how many backs you can stab and arses you can lick. This culminated in the global 'Covid' coordination between the Wetiko possessed who pulled it off in all the different countries without a trace of empathy and compassion for their impact on humans. Our sight sense can see only holographic form and not the Field which connects holographic form. Therefore we perceive 'physical' objects with 'space' in between. In fact that 'space' is energy/consciousness operating on multiple frequencies. One of them is Wetiko and that connects the Cult psychopaths, those who submit to the psychopaths, and those who serve the psychopaths in the media operations of the world. Wetiko is Gates. Wetiko is the mask-wearing submissive. Wetiko is the fake journalist and 'fact-checker'. The Wetiko Field is coordinating the whole thing. Psychopaths, gofers, media operatives, 'anti-hate' hate groups, 'fact-checkers' and submissive people work as one unit *even without human coordination* because they are attached to the *same* Field which is organising it all (Fig 22). Paul Levy is here describing how Wetiko-possessed people are drawn together and refuse to let any information breach their rigid

perceptions. He was writing long before 'Covid', but I think you will recognise followers of the 'Covid' religion *oh just a little bit*:

> People who are channelling the vibratory frequency of wetiko align with each other through psychic resonance to reinforce their unspoken shared agreement so as to uphold their deranged view of reality. Once an unconscious content takes possession of certain individuals, it irresistibly draws them together by mutual attraction and knits them into groups tied together by their shared madness that can easily swell into an avalanche of insanity.
>
> A psychic epidemic is a closed system, which is to say that it is insular and not open to any new information or informing influences from the outside world which contradict its fixed, limited, and limiting perspective.

There we have the Woke mind and the 'Covid' mind. Compatible resonance draws the awakening together, too, which is clearly happening today.

Figure 22: The Wetiko Field from which the Cult pyramid and its personnel are made manifest. (Image by Neil Hague).

Spiritual servitude

Wetiko doesn't care about humans. It's not human; it just possesses humans for its own ends and the effect (depending on the scale of

possession) can be anything from extreme psychopathy to unquestioning obedience. Wetiko's worst nightmare is for human consciousness to expand beyond the simulation. Everything is focussed on stopping that happening through control of information, thus perception, thus frequency. The 'education system', media, science, medicine, academia, are all geared to maintaining humanity in five-sense servitude as is the constant stimulation of low-vibrational mental and emotional states (see 'Covid'). Wetiko seeks to dominate those subconscious spaces between five-sense perception and expanded consciousness where the computer meets the operator. From these subconscious hiding places Wetiko speaks to us to trigger urges and desires that we take to be our own and manipulate us into anything from low-vibrational to psychopathic states. Remember how Islam describes the Jinn as invisible tricksters that 'whisper' and confuse. Wetiko is the origin of the 'trickster god' theme that you find in cultures all over the world. Jinn, like the Archons, are Wetiko which is terrified of humans awakening and reconnecting with our true self for then its energy source has gone. With that the feedback loop breaks between Wetiko and human perception that provides the energetic momentum on which its very existence depends as a force of evil. Humans are both its target and its source of survival, but only if we are operating in low-vibrational states of fear, hate, depression and the background anxiety that most people suffer. We are Wetiko's target because we are its key to survival. It needs us, not the other way round. Paul Levy writes:

> A vampire has no intrinsic, independent, substantial existence in its own right; it only exists in relation to us. The pathogenic, vampiric mind-parasite called wetiko is nothing in itself – not being able to exist from its own side – yet it has a 'virtual reality' such that it can potentially destroy our species …
>
> …The fact that a vampire is not reflected by a mirror can also mean that what we need to see is that there's nothing, no-thing to see, other than ourselves. The fact that wetiko is the expression of something inside of us means that the cure for wetiko is with us as well. The critical issue is finding this cure within us and then putting it into effect.

Evil begets evil because if evil does not constantly expand and find new sources of energetic sustenance its evil, its *distortion*, dies with the assimilation into balance and harmony. Love is the garlic to Wetiko's vampire. Evil, the absence of love, cannot exist in the presence of love. I think I see a way out of here. I have emphasised so many times over the decades that the Archons/Wetiko and their Cult are not all powerful. *They are not*. I don't care how it looks even now *they are not*. I have not called them little boys in short trousers for effect. I have said it because it is true. Wetiko's insatiable desire for power over others is not a sign of its omnipotence, but its insecurity. Paul Levy writes: 'Due to the primal fear which ultimately drives it and which it is driven to cultivate, wetiko's body politic has an intrinsic and insistent need for centralising power and control so as to create imagined safety for itself.' *Yeeeeeees!* Exactly! Why does Wetiko want humans in an ongoing state of fear? Wetiko itself *is* fear and it is petrified of love. As evil is an absence of love, so love is an absence of fear. Love conquers all and *especially* Wetiko which *is* fear. Wetiko brought fear into the world when it wasn't here before. *Fear* was the 'fall', the fall into low-frequency ignorance and illusion – fear is **F**alse **E**motion **A**ppearing **R**eal. The simulation is driven and energised by fear because Wetiko/Yaldabaoth (fear) *are* the simulation. Fear is the absence of love and Wetiko is the absence of love.

Wetiko today

We can now view current events from this level of perspective. The 'Covid' hoax has generated momentous amounts of ongoing fear, anxiety, depression and despair which have empowered Wetiko. No wonder people like Gates have been the instigators when they are Wetiko incarnate and exhibit every trait of Wetiko in the extreme. See how cold and unemotional these people are like Gates and his cronies, how dead of eye they are. That's Wetiko. Sabbatians are Wetiko and everything they control including the World Health Organization, Big Pharma and the 'vaccine' makers, national 'health'

hierarchies, corporate media, Silicon Valley, the banking system, and the United Nations with its planned transformation into world government. All are controlled and possessed by the Wetiko distortion into distorting human society in its image. We are with this knowledge at the gateway to understanding the world. Divisions of race, culture, creed and sexuality are diversions to hide the real division between those possessed and influenced by Wetiko and those that are not. The 'Covid' hoax has brought both clearly into view. Human behaviour is not about race. Tyrants and dictatorships come in all colours and creeds. What unites the US president bombing the innocent and an African tribe committing genocide against another as in Rwanda? What unites them? *Wetiko*. All wars are Wetiko, all genocide is Wetiko, all hunger over centuries in a world of plenty is Wetiko. Children going to bed hungry, including in the West, is Wetiko. Cult-generated Woke racial divisions that focus on the body are designed to obscure the reality that divisions in behaviour are manifestations of mind, not body. Obsession with body identity and group judgement is a means to divert attention from the real source of behaviour – mind and perception. Conflict sown by the Woke both within themselves and with their target groups are Wetiko providing lunch for itself through still more agents of the division, chaos, and fear on which it feeds. The Cult is seeking to assimilate the entirety of humanity and all children and young people into the Wetiko frequency by manipulating them into states of fear and despair. Witness all the suicide and psychological unravelling since the spring of 2020. Wetiko psychopaths want to impose a state of unquestioning obedience to authority which is no more than a conduit for Wetiko to enforce its will and assimilate humanity into itself. It needs us to believe that resistance is futile when it fears resistance and even more so the game-changing non-cooperation with its impositions. It can use violent resistance for its benefit. Violent impositions and violent resistance are *both* Wetiko. The Power of Love with its Power of No will sweep Wetiko from our world. Wetiko and its Cult know that. They just don't want us to know.

AI Wetiko

This brings me to AI or artificial intelligence and something else Wetikos don't want us to know. What is AI *really*? I know about computer code algorithms and AI that learns from data input. These, however, are more diversions, the expeditionary force, for the real AI that they want to connect to the human brain as promoted by Silicon Valley Wetikos like Kurzweil. What is this AI? It is the frequency of *Wetiko*, the frequency of the Archons. The connection of AI to the human brain is the connection of the Wetiko frequency to create a Wetiko hive mind and complete the job of assimilation. The hive mind is planned to be controlled from Israel and China which are both 100 percent owned by Wetiko Sabbatians. The assimilation process has been going on minute by minute in the 'smart' era which fused with the 'Covid' era. We are told that social media is scrambling the minds of the young and changing their personality. This is true, but what is social media? Look more deeply at how it works, how it creates divisions and conflict, the hostility and cruelty, the targeting of people until they are destroyed. That's Wetiko. Social media is manipulated to tune people to the Wetiko frequency with all the emotional exploitation tricks employed by platforms like Facebook and its Wetiko front man, Zuckerberg. Facebook's Instagram announced a new platform for children to overcome a legal bar on them using the main site. This is more Wetiko exploitation and manipulation of kids. Amnesty International likened the plan to foxes offering to guard the henhouse and said it was incompatible with human rights. Since when did Wetiko or Zuckerberg (I repeat myself) care about that? Would Brin and Page at Google, Wojcicki at YouTube, Bezos at Amazon and whoever the hell runs Twitter act as they do if they were not channelling Wetiko? Would those who are developing technologies for no other reason than human control? How about those designing and selling technologies to kill people and Big Pharma drug and 'vaccine' producers who know they will end or devastate lives? Quite a thought for these people to consider is that if you are Wetiko in a human life you are Wetiko on the 'other side' unless your frequency

changes and that can only change by a change of perception which becomes a change of behaviour. Where Gates is going does not bear thinking about although perhaps that's exactly where he wants to go. Either way, that's where he's going. His frequency will make it so.

The frequency lair

I have been saying for a long time that a big part of the addiction to smartphones and devices is that a frequency is coming off them that entraps the mind. People spend ages on their phones and sometimes even a minute or so after they put them down they pick them up again and it all repeats. 'Covid' lockdowns will have increased this addiction a million times for obvious reasons. Addictions to alcohol overindulgence and drugs are another way that Wetiko entraps consciousness to attach to its own. Both are symptoms of low-vibrational psychological distress which alcoholism and drug addiction further compound. Do we think it's really a coincidence that access to them is made so easy while potions that can take people into realms beyond the simulation are banned and illegal? I have explored smartphone addiction in other books, the scale is mind-blowing, and that level of addiction does not come without help. Tech companies that make these phones are Wetiko and they will have no qualms about destroying the minds of children. We are seeing again with these companies the Wetiko perceptual combination of psychopathic enforcers and weak and meek unquestioning compliance by the rank and file.

The global Smart Grid is the Wetiko Grid and it is crucial to complete the Cult endgame. The simulation is radiation and we are being deluged with technological radiation on a devastating scale. Wetiko frauds like Elon Musk serve Cult interests while occasionally criticising them to maintain his street-cred. 5G and other forms of Wi-Fi are being directed at the earth from space on a volume and scale that goes on increasing by the day. Elon Musk's (officially) SpaceX Starlink project is in the process of putting tens of thousands of satellites in low orbit to cover every inch of the planet with 5G and other Wi-Fi to create Kurzweil's global 'cloud' to which the

human mind is planned to be attached very soon. SpaceX has approval to operate 12,000 satellites with more than 1,300 launched at the time of writing and applications filed for *30,000* more. Other operators in the Wi-Fi, 5G, low-orbit satellite market include OneWeb (UK), Telesat (Canada), and AST & Science (US). Musk tells us that AI could be the end of humanity and then launches a company called Neuralink to connect the human brain to computers. Musk's (in theory) Tesla company is building electric cars and the driverless vehicles of the smart control grid. As frauds and bullshitters go Elon Musk in my opinion is Major League.

5G and technological radiation in general are destructive to human health, genetics and psychology and increasing the strength of artificial radiation underpins the five-sense perceptual bubbles which are themselves expressions of radiation or electromagnetism. Freedom activist John Whitehead was so right with his 'databit by databit, we are building our own electronic concentration camps'. The Smart Grid and 5G is a means to control the human mind and infuse perceptual information into The Field to influence anyone in sync with its frequency. You can change perception and behaviour en masse if you can manipulate the population into those levels of frequency and this is happening all around us today. The arrogance of Musk and his fellow Cult operatives knows no bounds in the way that we see with Gates. Musk's satellites are so many in number already they are changing the night sky when viewed from Earth. The astronomy community has complained about this and they have seen nothing yet. Some consequences of Musk's Wetiko hubris include: Radiation; visible pollution of the night sky; interference with astronomy and meteorology; ground and water pollution from intensive use of increasingly many spaceports; accumulating space debris; continual deorbiting and burning up of aging satellites, polluting the atmosphere with toxic dust and smoke; and ever-increasing likelihood of collisions. A collective public open letter of complaint to Musk said:

We are writing to you … because SpaceX is in process of surrounding the Earth with a network of thousands of satellites whose very purpose is to irradiate every square inch of the

Earth. SpaceX, like everyone else, is treating the radiation as if it were not there. As if the mitochondria in our cells do not depend on electrons moving undisturbed from the food we digest to the oxygen we breathe.

As if our nervous systems and our hearts are not subject to radio frequency interference like any piece of electronic equipment. As if the cancer, diabetes, and heart disease that now afflict a majority of the Earth's population are not metabolic diseases that result from interference with our cellular machinery. As if insects everywhere, and the birds and animals that eat them, are not starving to death as a result.

People like Musk and Gates believe in their limitless Wetiko arrogance that they can do whatever they like to the world because they own it. Consequences for humanity are irrelevant. It's absolutely time that we stopped taking this shit from these self-styled masters of the Earth when you consider where this is going.

Why is the Cult so anti-human?

I hear this question often: Why would they do this when it will affect them, too? Ah, but will it? Who is this *them*? Forget their bodies. They are just vehicles for Wetiko consciousness. When you break it all down to the foundations we are looking at a state of severely distorted consciousness targeting another state of consciousness for assimilation. The rest is detail. The simulation is the fly-trap in which unique sensations of the five senses create a cycle of addiction called reincarnation. Renegade Minds see that everything which happens in our reality is a smaller version of the whole picture in line with the holographic principle. Addiction to the radiation of smart technology is a smaller version of addiction to the whole simulation. Connecting the body/brain to AI is taking that addiction on a giant step further to total ongoing control by assimilating human incarnate consciousness into Wetiko. I have watched during the 'Covid' hoax how many are becoming ever more profoundly attached to Wetiko's perceptual calling cards of aggressive response to any other point of view ('There is no other god but me'), psychopathic lack of compassion and empathy, and servile submission to the narrative and will of authority. Wetiko is the psychopaths *and* subservience to psychopaths. The Cult of Wetiko is

so anti-human because it is *not* human. It embarked on a mission to destroy human by targeting everything that it means to be human and to survive as human. 'Covid' is not the end, just a means to an end. The Cult with its Wetiko consciousness is seeking to change Earth systems, including the atmosphere, to suit them, not humans. The gathering bombardment of 5G alone from ground and space is dramatically changing The Field with which the five senses interact. There is so much more to come if we sit on our hands and hope it will all go away. It is not meant to go away. It is meant to get ever more extreme and we need to face that while we still can – just.

Carbon dioxide is the gas of life. Without that human is over. Kaput, gone, history. No natural world, no human. The Cult has created a cock and bull story about carbon dioxide and climate change to justify its reduction to the point where Gates and the ignoramus Biden 'climate chief' John Kerry want to suck it out of the atmosphere. Kerry wants to do this because his master Gates does. Wetikos have made the gas of life a demon with the usual support from the Wokers of Extinction Rebellion and similar organisations and the bewildered puppet-child that is Greta Thunberg who was put on the world stage by Klaus Schwab and the World Economic Forum. The name Extinction Rebellion is both ironic and as always Wetiko inversion. The gas that we need to survive must be reduced to save us from extinction. The most basic need of human is oxygen and we now have billions walking around in face nappies depriving body and brain of this essential requirement of human existence. More than that 5G at 60 gigahertz interacts with the oxygen molecule to reduce the amount of oxygen the body can absorb into the bloodstream. The obvious knock-on consequences of that for respiratory and cognitive problems and life itself need no further explanation. Psychopaths like Musk are assembling a global system of satellites to deluge the human atmosphere with this insanity. The man should be in jail. Here we have two most basic of human needs, oxygen and carbon dioxide, being dismantled.

Two others, water and food, are getting similar treatment with the United Nations Agendas 21 and 2030 – the Great Reset – planning to

centrally control all water and food supplies. People will not even own rain water that falls on their land. Food is affected at the most basic level by reducing carbon dioxide. We have genetic modification or GMO infiltrating the food chain on a mass scale, pesticides and herbicides polluting the air and destroying the soil. Freshwater fish that provide livelihoods for 60 million people and feed hundreds of millions worldwide are being 'pushed to the brink' according the conservationists while climate change is the only focus. Now we have Gates and Schwab wanting to dispense with current food sources all together and replace them with a synthetic version which the Wetiko Cult would control in terms of production and who eats and who doesn't. We have been on the Totalitarian Tiptoe to this for more than 60 years as food has become ever more processed and full of chemical shite to the point today when it's not natural food at all. As Dr Tom Cowan says: 'If it has a label don't eat it.' Bill Gates is now the biggest owner of farmland in the United States and he does nothing without an ulterior motive involving the Cult. Klaus Schwab wrote: 'To feed the world in the next 50 years we will need to produce as much food as was produced in the last 10,000 years … food security will only be achieved, however, if regulations on genetically modified foods are adapted to reflect the reality that gene editing offers a precise, efficient and safe method of improving crops.' Liar. People and the world are being targeted with aluminium through vaccines, chemtrails, food, drink cans, and endless other sources when aluminium has been linked to many health issues including dementia which is increasing year after year. Insects, bees and wildlife essential to the food chain are being deleted by pesticides, herbicides and radiation which 5G is dramatically increasing with 6G and 7G to come. The pollinating bee population is being devastated while wildlife including birds, dolphins and whales are having their natural radar blocked by the effects of ever-increasing radiation. In the summer windscreens used to be splattered with insects so numerous were they. It doesn't happen now. Where have they gone?

Synthetic everything

The Cult is introducing genetically-modified versions of trees, plants and insects including a Gates-funded project to unleash hundreds of millions of genetically-modified, lab-altered and patented male mosquitoes to mate with wild mosquitoes and induce genetic flaws that cause them to die out. Clinically-insane Gates-funded Japanese researchers have developed mosquitos that spread vaccine and are dubbed 'flying vaccinators'. Gates is funding the modification of weather patterns in part to sell the myth that this is caused by carbon dioxide and he's funding geoengineering of the skies to change the atmosphere. Some of this came to light with the Gates-backed plan to release tonnes of chalk into the atmosphere to 'deflect the Sun and cool the planet'. Funny how they do this while the heating effect of the Sun is not factored into climate projections focussed on carbon dioxide. The reason is that they want to reduce carbon dioxide (so don't mention the Sun), but at the same time they do want to reduce the impact of the Sun which is so essential to human life and health. I have mentioned the sun-cholesterol-vitamin D connection as they demonise the Sun with warnings about skin cancer (caused by the chemicals in sun cream they tell you to splash on). They come from the other end of the process with statin drugs to reduce cholesterol that turns sunlight into vitamin D. A lack of vitamin D leads to a long list of health effects and how vitamin D levels must have fallen with people confined to their homes over 'Covid'. Gates is funding other forms of geoengineering and most importantly chemtrails which are dropping heavy metals, aluminium and self-replicating nanotechnology onto the Earth which is killing the natural world. See *Everything You Need To Know, But Have Never Been Told* for the detailed background to this.

Every human system is being targeted for deletion by a force that's not human. The Wetiko Cult has embarked on the process of transforming the human body from biological to synthetic biological as I have explained. Biological is being replaced by the artificial and synthetic – Archontic 'countermimicry' – right across human society. The plan eventually is to dispense with the human body altogether

and absorb human consciousness – which it wouldn't really be by then – into cyberspace (the simulation which is Wetiko/Yaldabaoth). Preparations for that are already happening if people would care to look. The alternative media rightly warns about globalism and 'the globalists', but this is far bigger than that and represents the end of the human race as we know it. The 'bad copy' of prime reality that Gnostics describe was a bad copy of harmony, wonder and beauty to start with before Wetiko/Yaldabaoth set out to change the simulated 'copy' into something very different. The process was slow to start with. Entrapped humans in the simulation timeline were not technologically aware and they had to be brought up to intellectual speed while being suppressed spiritually to the point where they could build their own prison while having no idea they were doing so. We have now reached that stage where technological intellect has the potential to destroy us and that's why events are moving so fast. Central American shaman Don Juan Matus said:

> Think for a moment, and tell me how you would explain the contradictions between the intelligence of man the engineer and the stupidity of his systems of belief, or the stupidity of his contradictory behaviour. Sorcerers believe that the predators have given us our systems of beliefs, our ideas of good and evil; our social mores. They are the ones who set up our dreams of success or failure. They have given us covetousness, greed, and cowardice. It is the predator who makes us complacent, routinary, and egomaniacal.
>
> In order to keep us obedient and meek and weak, the predators engaged themselves in a stupendous manoeuvre – stupendous, of course, from the point of view of a fighting strategist; a horrendous manoeuvre from the point of those who suffer it. They gave us their mind. The predators' mind is baroque, contradictory, morose, filled with the fear of being discovered any minute now.

For 'predators' see Wetiko, Archons, Yaldabaoth, Jinn, and all the other versions of the same phenomenon in cultures and religions all over the world. The theme is always the same because it's true and it's real. We have reached the point where we have to deal with it. The question is – how?

Don't fight – walk away

I thought I'd use a controversial subheading to get things moving in terms of our response to global fascism. What do you mean 'don't fight'? What do you mean 'walk away'? We've got to fight. We can't walk away. Well, it depends what we mean by fight and walk away. If fighting means physical combat we are playing Wetiko's game and falling for its trap. It wants us to get angry, aggressive, and direct hate and hostility at the enemy we think we must fight. Every war, every battle, every conflict, has been fought with Wetiko leading both sides. It's what it does. Wetiko wants a fight, anywhere, any place. Just hit me, son, so I can hit you back. Wetiko hits Wetiko and Wetiko hits Wetiko in return. I am very forthright as you can see in exposing Wetikos of the Cult, but I don't hate them. I refuse to hate them. It's what they want. What you hate you become. What you *fight* you become. Wokers, 'anti-haters' and 'anti-fascists' prove this every time they reach for their keyboards or don their balaclavas. By walk away I mean to disengage from Wetiko which includes ceasing to cooperate with its tyranny. Paul Levy says of Wetiko:

> The way to 'defeat' evil is not to try to destroy it (for then, in playing evil's game, we have already lost), but rather, to find the invulnerable place within ourselves where evil is unable to vanquish us – this is to truly 'win' our battle with evil.

Wetiko is everywhere in human society and it's been on steroids since the 'Covid' hoax. Every shouting match over wearing masks has Wetiko wearing a mask and Wetiko not wearing one. It's an electrical circuit of push and resist, push and resist, with Wetiko pushing *and* resisting. Each polarity is Wetiko empowering itself. Dictionary definitions of 'resist' include 'opposing, refusing to accept or comply with' and the word to focus on is 'opposing'. What form does this take – setting police cars alight or 'refusing to accept or comply with'? The former is Wetiko opposing Wetiko while the other points the way forward. This is the difference between those aggressively demanding that government fascism must be obeyed who stand in stark contrast to the great majority of Pushbackers. We saw this clearly with a march by thousands of Pushbackers against lockdown in London followed days later by a Woker-hijacked

protest in Bristol in which police cars were set on fire. Masks were virtually absent in London and widespread in Bristol. Wetiko wants lockdown on every level of society and infuses its aggression to police it through its unknowing stooges. Lockdown protesters are the ones with the smiling faces and the hugs, The two blatantly obvious states of being – getting more obvious by the day – are the result of Wokers and their like becoming ever more influenced by the simulation Field of Wetiko and Pushbackers ever more influenced by The Field of a far higher vibration beyond the simulation. Wetiko can't invade the heart which is where most lockdown opponents are coming from. It's the heart that allows them to see through the lies to the truth in ways I will be highlighting.

Renegade Minds know that calmness is the place from which wisdom comes. You won't find wisdom in a hissing fit and wisdom is what we need in abundance right now. Calmness is not weakness – you don't have to scream at the top of your voice to be strong. Calmness is indeed a sign of strength. 'No' means I'm not doing it. *NOOOO!!!* doesn't mean you're not doing it even more. Volume does not advance 'No – I'm not doing it'. You are just not doing it. Wetiko possessed and influenced don't know how to deal with that. Wetiko wants a fight and we should not give it one. What it needs more than anything is our *cooperation* and we should not give that either. Mass rallies and marches are great in that they are a visual representation of feeling, but if it ends there they are irrelevant. You demand that Wetikos act differently? Well, they're not going to are they? They are Wetikos. We don't need to waste our time demanding that something doesn't happen when that will make no difference. We need to delete the means that *allows* it to happen. This, invariably, is our cooperation. You can demand a child stop firing a peashooter at the dog or you can refuse to buy the peashooter. If you provide the means you are cooperating with the dog being smacked on the nose with a pea. How can the authorities enforce mask-wearing if millions in a country refuse? What if the 74 million Pushbackers that voted for Trump in 2020 refused to wear masks, close their businesses or stay in their homes. It would be unenforceable. The

few control the many through the compliance of the many and that's always been the dynamic be it 'Covid' regulations or the Roman Empire. I know people can find it intimidating to say no to authority or stand out in a crowd for being the only one with a face on display; but it has to be done or it's over. I hope I've made clear in this book that where this is going will be far more intimidating than standing up now and saying 'No' – I will not cooperate with my own enslavement and that of my children. There might be consequences for some initially, although not so if enough do the same. The question that must be addressed is what is going to happen if we don't? It is time to be strong and unyieldingly so. No means no. Not here and there, but *everywhere* and *always*. I have refused to wear a mask and obey all the other nonsense. I will not comply with tyranny. I repeat: Fascism is not imposed by fascists – there are never enough of them. Fascism is imposed by the population acquiescing to fascism. *I will not do it*. I will die first, or my body will. Living meekly under fascism is a form of death anyway, the death of the spirit that Martin Luther King described.

Making things happen

We must not despair. This is not over till it's over and it's far from that. The 'fat lady' must refuse to sing. The longer the 'Covid' hoax has dragged on and impacted on more lives we have seen an awakening of phenomenal numbers of people worldwide to the realisation that what they have believed all their lives is not how the world really is. Research published by the system-serving University of Bristol and King's College London in February, 2021, concluded: 'One in every 11 people in Britain say they trust David Icke's take on the coronavirus pandemic.' It will be more by now and we have gathering numbers to build on. We must urgently progress from seeing the scam to ceasing to cooperate with it. Prominent German lawyer Reiner Fuellmich, also licenced to practice law in America, is doing a magnificent job taking the legal route to bring the psychopaths to justice through a second Nuremberg tribunal for crimes against humanity. Fuellmich has an impressive record of

beating the elite in court and he formed the German Corona Investigative Committee to pursue civil charges against the main perpetrators with a view to triggering criminal charges. Most importantly he has grasped the foundation of the hoax – the PCR test not testing for the 'virus' – and Christian Drosten is therefore on his charge sheet along with Gates frontman Tedros at the World Health Organization. Major players must be not be allowed to inflict their horrors on the human race without being brought to book. A life sentence must follow for Bill Gates and the rest of them. A group of researchers has also indicted the government of Norway for crimes against humanity with copies sent to the police and the International Criminal Court. The lawsuit cites participation in an internationally-planned false pandemic and violation of international law and human rights, the European Commission's definition of human rights by coercive rules, Nuremberg and Hague rules on fundamental human rights, and the Norwegian constitution. We must take the initiative from hereon and not just complain, protest and react.

There are practical ways to support vital mass non-cooperation. Organising in numbers is one. Lockdown marches in London in the spring in 2021 were mass non-cooperation that the authorities could not stop. There were too many people. Hundreds of thousands walked the London streets in the centre of the road for mile after mile while the Face-Nappies could only look on. They were determined, but calm, and just *did it* with no histrionics and lots of smiles. The police were impotent. Others are organising group shopping without masks for mutual support and imagine if that was happening all over. Policing it would be impossible. If the store refuses to serve people in these circumstances they would be faced with a long line of trolleys full of goods standing on their own and everything would have to be returned to the shelves. How would they cope with that if it kept happening? I am talking here about moving on from complaining to being pro-active; from watching things happen to making things happen. I include in this our relationship with the police. The behaviour of many Face-Nappies

has been disgraceful and anyone who thinks they would never find concentration camp guards in the 'enlightened' modern era have had that myth busted big-time. The period and setting may change – Wetikos never do. I watched film footage from a London march in which a police thug viciously kicked a protestor on the floor who had done nothing. His fellow Face-Nappies stood in a ring protecting him. What he did was a criminal assault and with a crowd far outnumbering the police this can no longer be allowed to happen unchallenged. I get it when people chant 'shame on you' in these circumstances, but that is no longer enough. They *have* no shame those who do this. Crowds needs to start making a citizen's arrest of the police who commit criminal offences and brutally attack innocent people and defenceless women. A citizen's arrest can be made under section 24A of the UK Police and Criminal Evidence (PACE) Act of 1984 and you will find something similar in other countries. I prefer to call it a Common Law arrest rather than citizen's for reasons I will come to shortly. Anyone can arrest a person committing an indictable offence or if they have reasonable grounds to suspect they are committing an indictable offence. On both counts the attack by the police thug would have fallen into this category. A citizen's arrest can be made to stop someone:

- Causing physical injury to himself or any other person
- Suffering physical injury
- Causing loss of or damage to property
- Making off before a constable can assume responsibility for him

A citizen's arrest may also be made to prevent a breach of the peace under Common Law and if they believe a breach of the peace will happen or anything related to harm likely to be done or already done in their presence. This is the way to go I think – the Common Law version. If police know that the crowd and members of the public will no longer be standing and watching while they commit

their thuggery and crimes they will think twice about acting like Brownshirts and Blackshirts.

Common Law – common sense

Mention of Common Law is very important. Most people think the law is the law as in one law. This is not the case. There are two bodies of law, Common Law and Statute Law, and they are not the same. Common Law is founded on the simple premise of do no harm. It does not recognise victimless crimes in which no harm is done while Statute Law does. There is a Statute Law against almost everything. So what is Statute Law? Amazingly it's the law of the *sea* that was brought ashore by the Cult to override the law of the land which is Common Law. They had no right to do this and as always they did it anyway. They had to. They could not impose their will on the people through Common Law which only applies to do no harm. How could you stitch up the fine detail of people's lives with that? Instead they took the law of the sea, or Admiralty Law, and applied it to the population. Statute Law refers to all the laws spewing out of governments and their agencies including all the fascist laws and regulations relating to 'Covid'. The key point to make is that Statute Law is *contract law*. It only applies between *contracting* corporations. Most police officers don't even know this. They have to be kept in the dark, too. Long ago when merchants and their sailing ships began to trade with different countries a contractual law was developed called Admiralty Law and other names. Again it only applied to *contracts* agreed between *corporate* entities. If there is no agreed contract the law of the sea had no jurisdiction *and that still applies to its new alias of Statute Law*. The problem for the Cult when the law of the sea was brought ashore was an obvious one. People were not corporations and neither were government entities. To overcome the latter they made governments and all associated organisations corporations. All the institutions are *private corporations* and I mean governments and their agencies, local councils, police, courts, military, US states, the whole lot. Go to the

Dun and Bradstreet corporate listings website for confirmation that they are all corporations. You are arrested by a private corporation called the police by someone who is really a private security guard and they take you to court which is another private corporation. Neither have jurisdiction over you unless you consent and *contract* with them. This is why you hear the mantra about law enforcement policing by *consent* of the people. In truth the people 'consent' only in theory through monumental trickery.

Okay, the Cult overcame the corporate law problem by making governments and institutions corporate entities; but what about people? They are not corporations are they? Ah ... well in a sense, and *only* a sense, they are. Not people exactly – the illusion of people. The Cult creates a corporation in the name of everyone at the time that their birth certificate is issued. Note birth/ *berth* certificate and when you go to court under the law of the sea on land you stand in a *dock*. These are throwbacks to the origin. My Common Law name is David Vaughan Icke. The name of the corporation created by the government when I was born is called Mr David Vaughan Icke usually written in capitals as MR DAVID VAUGHAN ICKE. That is not me, the living, breathing man. It is a fictitious corporate entity. The trick is to make you think that David Vaughan Icke and MR DAVID VAUGHAN ICKE are the same thing. *They are not*. When police charge you and take you to court they are prosecuting the corporate entity and not the living, breathing, man or woman. They have to trick you into identifying as the corporate entity and contracting with them. Otherwise they have no jurisdiction. They do this through a language known as legalese. Lawful and legal are not the same either. Lawful relates to Common Law and legal relates to Statute Law. Legalese is the language of Statue Law which uses terms that mean one thing to the public and another in legalese. Notice that when a police officer tells someone why they are being charged he or she will say at the end: 'Do you understand?' To the public that means 'Do you comprehend?' In legalese it means 'Do you stand under me?' Do you stand under my authority? If you say

yes to the question you are unknowingly agreeing to give them jurisdiction over you in a contract between two corporate entities.

This is a confidence trick in every way. Contracts have to be agreed between informed parties and if you don't know that David Vaughan Icke is agreeing to be the corporation MR DAVID VAUGHAN ICKE you cannot knowingly agree to contract. They are deceiving you and another way they do this is to ask for proof of identity. You usually show them a driving licence or other document on which your corporate name is written. In doing so you are accepting that you are that corporate entity when you are not. Referring to yourself as a 'person' or 'citizen' is also identifying with your corporate fiction which is why I made the Common Law point about the citizen's arrest. If you are approached by a police officer you identify yourself immediately as a living, breathing, man or woman and say 'I do not consent, I do not contract with you and I do not understand' or stand under their authority. I have a Common Law birth certificate as a living man and these are available at no charge from commonlawcourt.com. Businesses registered under the Statute Law system means that its laws apply. There are, however, ways to run a business under Common Law. Remember all 'Covid' laws and regulations are Statute Law – the law of *contracts* and you do not have to contract. This doesn't mean that you can kill someone and get away with it. Common Law says do no harm and that applies to physical harm, financial harm etc. Police are employees of private corporations and there needs to be a new system of non-corporate Common Law constables operating outside the Statute Law system. If you go to davidicke.com and put Common Law into the search engine you will find videos that explain Common Law in much greater detail. It is definitely a road we should walk.

With all my heart

I have heard people say that we are in a spiritual war. I don't like the term 'war' with its Wetiko dynamic, but I know what they mean. Sweep aside all the bodily forms and we are in a situation in which two states of consciousness are seeking very different realities.

Wetiko wants upheaval, chaos, fear, suffering, conflict and control. The other wants love, peace, harmony, fairness and freedom. That's where we are. We should not fall for the idea that Wetiko is all-powerful and there's nothing we can do. Wetiko is not all-powerful. It's a joke, pathetic. It doesn't have to be, but it has made that choice for now. A handful of times over the years when I have felt the presence of its frequency I have allowed it to attach briefly so I could consciously observe its nature. The experience is not pleasant, the energy is heavy and dark, but the ease with which you can kick it back out the door shows that its real power is in persuading us that it has power. It's all a con. Wetiko is a con. It's a trickster and not a power that can control us if we unleash our own. The con is founded on manipulating humanity to give its power to Wetiko which recycles it back to present the illusion that it has power when its power is *ours* that we gave away. This happens on an energetic level and plays out in the world of the seen as humanity giving its power to Wetiko authority which uses that power to control the population when the power is only the power the population has handed over. How could it be any other way for billions to be controlled by a relative few? I have had experiences with people possessed by Wetiko and again you can kick its arse if you do it with an open heart. Oh yes – the *heart* which can transform the world of perceived 'matter'.

We are receiver-transmitters and processors of information, but what information and where from? Information is processed into perception in three main areas – the brain, the heart and the belly. These relate to thinking, knowing, and emotion. Wetiko wants us to be head and belly people which means we think within the confines of the Matrix simulation and low-vibrational emotional reaction scrambles balance and perception. A few minutes on social media and you see how emotion is the dominant force. Woke is all emotion and is therefore thought-free and fact-free. Our heart is something different. It *knows* while the head *thinks* and has to try to work it out because it doesn't know. The human energy field has seven prime vortexes which connect us with wider reality (Fig 23). Chakra means

'wheels of light' in the Sanskrit language of ancient India. The main ones are: The crown chakra on top of the head; brow (or 'third eye') chakra in the centre of the forehead; throat chakra; heart chakra in the centre of the chest; solar plexus chakra below the sternum; sacral chakra beneath the navel; and base chakra at the bottom of the spine. Each one has a particular function or functions. We feel anxiety and nervousness in the belly where the sacral chakra is located and this processes emotion that can affect the colon to give people 'the shits' or make them 'shit scared' when they are nervous. Chakras all play an important role, but the Mr and Mrs Big is the heart chakra which sits at the centre of the seven, above the chakras that connect us to the 'physical' and below those that connect with higher realms (or at least should). Here in the heart chakra we feel love, empathy and compassion – 'My heart goes out to you'. Those with closed hearts become literally 'heart-less' in their attitudes and behaviour (see Bill Gates). Native Americans portrayed Wetiko with what Paul Levy calls a 'frigid, icy heart, devoid of mercy' (see Bill Gates).

Figure 23: The chakra system which interpenetrates the human energy field. The heart chakra is the governor – or should be.

Wetiko trembles at the thought of heart energy which it cannot infiltrate. The frequency is too high. What it seeks to do instead is close the heart chakra vortex to block its perceptual and energetic influence. Psychopaths have 'hearts of stone' and emotionally-damaged people have 'heartache' and 'broken hearts'. The astonishing amount of heart disease is related to heart chakra

disruption with its fundamental connection to the 'physical' heart. Dr Tom Cowan has written an outstanding book challenging the belief that the heart is a pump and making the connection between the 'physical' and spiritual heart. Rudolph Steiner who was way ahead of his time said the same about the fallacy that the heart is a pump. *What*? The heart is not a pump? That's crazy, right? Everybody knows that. Read Cowan's *Human Heart, Cosmic Heart* and you will realise that the very idea of the heart as a pump is ridiculous when you see the evidence. How does blood in the feet so far from the heart get pumped horizontally up the body by the heart?? Cowan explains in the book the real reason why blood moves as it does. Our 'physical' heart is used to symbolise love when the source is really the heart vortex or spiritual heart which is our most powerful energetic connection to 'out there' expanded consciousness. That's why we feel *knowing* – intuitive knowing – in the centre of the chest. Knowing doesn't come from a process of thoughts leading to a conclusion. It is there in an instant all in one go. Our heart knows because of its connection to levels of awareness that *do* know. This is the meaning and source of intuition – intuitive *knowing*.

For the last more than 30 years of uncovering the global game and the nature of reality my heart has been my constant antenna for truth and accuracy. An American intelligence insider once said that I had quoted a disinformer in one of my books and yet I had only quoted the part that was true. He asked: 'How do you do that?' By using my heart antenna was the answer and anyone can do it. Heart-centred is how we are meant to be. With a closed heart chakra we withdraw into a closed mind and the bubble of five-sense reality. If you take a moment to focus your attention on the centre of your chest, picture a spinning wheel of light and see it opening and expanding. You will feel it happening, too, and perceptions of the heart like joy and love as the heart impacts on the mind as they interact. The more the chakra opens the more you will feel expressions of heart consciousness and as the process continues, and becomes part of you, insights and knowings will follow. An open

heart is connected to that level of awareness that knows all is *One*. You will see from its perspective that the fault-lines that divide us are only illusions to control us. An open heart does not process the illusions of race, creed and sexuality except as brief experiences for a consciousness that is all. Our heart does not see division, only unity (Figs 24 and 25). There's something else, too. Our hearts love to laugh. Mark Twain's quote that says 'The human race has one really effective weapon, and that is laughter' is really a reference to the heart which loves to laugh with the joy of knowing the true nature of infinite reality and that all the madness of human society is an illusion of the mind. Twain also said: 'Against the assault of laughter nothing can stand.' This is so true of Wetiko and the Cult. Their insecurity demands that they be taken seriously and their power and authority acknowledged and feared. We should do nothing of the sort. We should not get aggressive or fearful which their insecurity so desires. We should laugh in their face. Even in their no-face as police come over in their face-nappies and expect to be taken seriously. They don't take themselves seriously looking like that so why should we? Laugh in the face of intimidation. Laugh in the face of tyranny. You will see by its reaction that you have pressed all of its buttons. Wetiko does not know what to do in the face of laughter or when its targets refuse to concede their joy to fear. We have seen many examples during the 'Covid' hoax when people have expressed their energetic power and the string puppets of Wetiko retreat with their tail limp between their knees. Laugh – the world is bloody mad after all and if it's a choice between laughter and tears I know which way I'm going.

Figure 24: Head consciousness without the heart sees division and everything apart from everything else.

Figure 25: Heart consciousness sees everything as One.

'Vaccines' and the soul

The foundation of Wetiko/Archon control of humans is the separation of incarnate five-sense mind from the infinite 'I' and closing the heart chakra where the True 'I' lives during a human life. The goal has been to achieve complete separation in both cases. I was interested therefore to read an account by a French energetic healer of what she said she experienced with a patient who had been given the 'Covid' vaccine. Genuine energy healers can sense information and consciousness fields at different levels of being which are referred to as 'subtle bodies'. She described treating the patient who later returned after having, without the healer's knowledge, two doses of the 'Covid vaccine'. The healer said:

I noticed immediately the change, very heavy energy emanating from [the] subtle bodies. The scariest thing was when I was working on the heart chakra, I connected with her soul: it was detached from the physical body, it had no contact and it was, as if it was floating in a state of total confusion: a damage to the consciousness that loses contact with the physical body, i.e. with our biological machine, there is no longer any communication between them.

I continued the treatment by sending light to the heart chakra, the soul of the person, but it seemed that the soul could no longer receive any light, frequency or energy. It was a very powerful experience for me. Then I understood that this substance is indeed used to detach consciousness so that this consciousness can no longer interact through this body that it possesses in life, where there is no longer any contact, no frequency, no light, no more energetic balance or mind.

This would create a human that is rudderless and at the extreme almost zombie-like operating with a fractional state of consciousness at the mercy of Wetiko. I was especially intrigued by what the healer said in the light of the prediction by the highly-informed Rudolf Steiner more than a hundred years ago. He said:

> In the future, we will eliminate the soul with medicine. Under the pretext of a 'healthy point of view', there will be a vaccine by which the human body will be treated as soon as possible directly at birth, so that the human being cannot develop the thought of the existence of soul and Spirit. To materialistic doctors will be entrusted the task of removing the soul of humanity.
>
> As today, people are vaccinated against this disease or that disease, so in the future, children will be vaccinated with a substance that can be produced precisely in such a way that people, thanks to this vaccination, will be immune to being subjected to the 'madness' of spiritual life. He would be extremely smart, but he would not develop a conscience, and that is the true goal of some materialistic circles.

Steiner said the vaccine would detach the physical body from the etheric body (subtle bodies) and 'once the etheric body is detached the relationship between the universe and the etheric body would become extremely unstable, and man would become an automaton'. He said 'the physical body of man must be polished on this Earth by spiritual will – so the vaccine becomes a kind of arymanique (Wetiko) force' and 'man can no longer get rid of a given materialistic feeling'. Humans would then, he said, become 'materialistic of constitution and can no longer rise to the spiritual'. I have been writing for years about DNA being a receiver-transmitter of information that connects us to other levels of reality and these 'vaccines' changing DNA can be likened to changing an antenna and what it can transmit and receive. Such a disconnection would clearly lead to changes in personality and perception. Steiner further predicted the arrival of AI. Big Pharma 'Covid vaccine' makers, expressions of Wetiko, are testing their DNA-manipulating evil on children as I write with a view to giving the 'vaccine' to babies. If it's a soul-body disconnector – and I say that it is or can be – every child would be disconnected from 'soul' at birth and the 'vaccine' would create a closed system in which spiritual guidance from the greater self would play no part. This has been the ambition of Wetiko all

along. A Pentagon video from 2005 was leaked of a presentation explaining the development of vaccines to change behaviour by their effect on the brain. Those that believe this is not happening with the 'Covid' genetically-modifying procedure masquerading as a 'vaccine' should make an urgent appointment with Naivety Anonymous. Klaus Schwab wrote in 2018:

> Neurotechnologies enable us to better influence consciousness and thought and to understand many activities of the brain. They include decoding what we are thinking in fine levels of detail through new chemicals and interventions that can influence our brains to correct for errors or enhance functionality.

The plan is clear and only the heart can stop it. With every heart that opens, every mind that awakens, Wetiko is weakened. Heart and love are far more powerful than head and hate and so nothing like a majority is needed to turn this around.

Beyond the Phantom

Our heart is the prime target of Wetiko and so it must be the answer to Wetiko. We *are* our heart which is part of one heart, the infinite heart. Our heart is where the true self lives in a human life behind firewalls of five-sense illusion when an imposter takes its place – *Phantom Self*; but our heart waits patiently to be set free any time we choose to see beyond the Phantom, beyond Wetiko. A Wetikoed Phantom Self can wreak mass death and destruction while the love of forever is locked away in its heart. The time is here to unleash its power and let it sweep away the fear and despair that is Wetiko. Heart consciousness does not seek manipulated, censored, advantage for its belief or religion, its activism and desires. As an expression of the One it treats all as One with the same rights to freedom and opinion. Our heart demands fairness for itself no more than for others. From this unity of heart we can come together in mutual support and transform this Wetikoed world into what reality is meant to be – a place of love, joy, happiness, fairness, justice and freedom. Wetiko has another agenda and that's why the world is as

it is, but enough of this nonsense. Wetiko can't stay where hearts are open and it works so hard to keep them closed. Fear is its currency and its food source and love in its true sense has no fear. Why would love have fear when it knows it is *All That Is, Has Been, And Ever Can Be* on an eternal exploration of all possibility? Love in this true sense is not the physical attraction that passes for love. This can be an expression of it, yes, but Infinite Love, a love without condition, goes far deeper to the core of all being. It *is* the core of all being. Infinite realty was born from love beyond the illusions of the simulation. Love infinitely expressed is the knowing that all is One and the swiftly-passing experience of separation is a temporary hallucination. You cannot disconnect from Oneness; you can only *perceive* that you have and withdraw from its influence. This is the most important of all perception trickery by the mind parasite that is Wetiko and the foundation of all its potential for manipulation.

If we open our hearts, open the sluice gates of the mind, and redefine self-identity amazing things start to happen. Consciousness expands or contracts in accordance with self-identity. When true self is recognised as infinite awareness and label self – Phantom Self – is seen as only a series of brief experiences life is transformed. Consciousness expands to the extent that self-identity expands and everything changes. You see unity, not division, the picture, not the pixels. From this we can play the long game. No more is an experience something in and of itself, but a fleeting moment in the eternity of forever. Suddenly people in uniform and dark suits are no longer intimidating. Doing what your heart knows to be right is no longer intimidating and consequences for those actions take on the same nature of a brief experience that passes in the blink of an infinite eye. Intimidation is all in the mind. Beyond the mind there is no intimidation.

An open heart does not consider consequences for what it knows to be right. To do so would be to consider not doing what it knows to be right and for a heart in its power that is never an option. The Renegade Mind is really the Renegade Heart. Consideration of consequences will always provide a getaway car for the mind and

the heart doesn't want one. What is right in the light of what we face today is to stop cooperating with Wetiko in all its forms and to do it without fear or compromise. You cannot compromise with tyranny when tyranny always demands more until it has everything. Life is your perception and you are your destiny. Change your perception and you change your life. Change collective perception and we change the world.

Come on people … One human family, One heart, One goal … *FREEEEEEDOM*!

We must settle for nothing less.

Postscript

The big scare story as the book goes to press is the 'Indian' variant and the world is being deluged with propaganda about the 'Covid catastrophe' in India which mirrors in its lies and misrepresentations what happened in Italy before the first lockdown in 2020.

The *New York Post* published a picture of someone who had 'collapsed in the street from Covid' in India in April, 2021, which was actually taken during a gas leak in May, 2020. Same old, same old. Media articles in mid-February were asking why India had been so untouched by 'Covid' and then as their vaccine rollout gathered pace the alleged 'cases' began to rapidly increase. Indian 'Covid vaccine' maker Bharat Biotech was funded into existence by the Bill and Melinda Gates Foundation (the pair announced their divorce in May, 2021, which is a pity because they so deserve each other). The Indian 'Covid crisis' was ramped up by the media to terrify the world and prepare people for submission to still more restrictions. The scam that worked the first time was being repeated only with far more people seeing through the deceit. Davidicke.com and Ickonic.com have sought to tell the true story of what is happening by talking to people living through the Indian nightmare which has nothing to do with 'Covid'. We posted a letter from 'Alisha' in Pune who told a very different story to government and media mendacity. She said scenes of dying people and overwhelmed hospitals were designed to hide what was really happening – genocide and starvation. Alisha said that millions had already died of starvation during the ongoing lockdowns while government and media were lying and making it look like the 'virus':

Restaurants, shops, gyms, theatres, basically everything is shut. The cities are ghost towns. Even so-called 'essential' businesses are only open till 11am in the morning. You basically have just an hour to buy food and then your time is up.

Inter-state travel and even inter-district travel is banned. The cops wait at all major crossroads to question why you are traveling outdoors or to fine you if you are not wearing a mask.

The medical community here is also complicit in genocide, lying about hospitals being full and turning away people with genuine illnesses, who need immediate care. They have even created a shortage of oxygen cylinders.

This is the classic Cult modus operandi played out in every country. Alisha said that people who would not have a PCR test not testing for the 'virus' were being denied hospital treatment. She said the people hit hardest were migrant workers and those in rural areas. Most businesses employed migrant workers and with everything closed there were no jobs, no income and no food. As a result millions were dying of starvation or malnutrition. All this was happening under Prime Minister Narendra Modi, a 100-percent asset of the Cult, and it emphasises yet again the scale of pure anti-human evil we are dealing with. Australia banned its people from returning home from India with penalties for trying to do so of up to five years in jail and a fine of £37,000. The manufactured 'Covid' crisis in India was being prepared to justify further fascism in the West. Obvious connections could be seen between the Indian 'vaccine' programme and increased 'cases' and this became a common theme. The Seychelles, the most per capita 'Covid vaccinated' population in the world, went back into lockdown after a 'surge of cases'.

Long ago the truly evil Monsanto agricultural biotechnology corporation with its big connections to Bill Gates devastated Indian farming with genetically-modified crops. Human rights activist Gurcharan Singh highlighted the efforts by the Indian government to complete the job by destroying the food supply to hundreds of millions with 'Covid' lockdowns. He said that 415 million people at the bottom of the disgusting caste system (still going whatever they say) were below the poverty line and struggled to feed themselves every year. Now the government was imposing lockdown at just the

time to destroy the harvest. This deliberate policy was leading to mass starvation. People may reel back at the suggestion that a government would do that, but Wetiko-controlled 'leaders' are capable of any level of evil. In fact what is described in India is in the process of being instigated worldwide. The food chain and food supply are being targeted at every level to cause world hunger and thus control. Bill Gates is not the biggest owner of farmland in America for no reason and destroying access to food aids both the depopulation agenda and the plan for synthetic 'food' already being funded into existence by Gates. Add to this the coming hyper-inflation from the suicidal creation of fake 'money' in response to 'Covid' and the breakdown of container shipping systems and you have a cocktail that can only lead one way and is meant to. The Cult plan is to crash the entire system to 'build back better' with the Great Reset.

'Vaccine' transmission

Reports from all over the world continue to emerge of women suffering menstrual and fertility problems after having the fake 'vaccine' and of the non-'vaccinated' having similar problems when interacting with the 'vaccinated'. There are far too many for 'coincidence' to be credible. We've had menopausal women getting periods, others having periods stop or not stopping for weeks, passing clots, sometimes the lining of the uterus, breast irregularities, and miscarriages (which increased by 400 percent in parts of the United States). Non-'vaccinated' men and children have suffered blood clots and nose bleeding after interaction with the 'vaccinated'. Babies have died from the effects of breast milk from a 'vaccinated' mother. Awake doctors – the small minority – speculated on the cause of non-'vaccinated' suffering the same effects as the 'vaccinated'. Was it nanotechnology in the synthetic substance transmitting frequencies or was it a straight chemical bioweapon that was being transmitted between people? I am not saying that some kind of chemical transmission is not one possible answer, but the foundation of all that the Cult does is frequency and

this is fertile ground for understanding how transmission can happen. American doctor Carrie Madej, an internal medicine physician and osteopath, has been practicing for the last 20 years, teaching medical students, and she says attending different meetings where the agenda for humanity was discussed. Madej, who operates out of Georgia, did not dismiss other possible forms of transmission, but she focused on frequency in search of an explanation for transmission. She said the Moderna and Pfizer 'vaccines' contained nano-lipid particles as a key component. This was a brand new technology never before used on humanity. 'They're using a nanotechnology which is pretty much little tiny computer bits … nanobots or hydrogel.' Inside the 'vaccines' was 'this sci-fi kind of substance' which suppressed immune checkpoints to get into the cell. I referred to this earlier as the 'Trojan horse' technique that tricks the cell into opening a gateway for the self-replicating synthetic material and while the immune system is artificially suppressed the body has no defences. Madej said the substance served many purposes including an on-demand ability to 'deliver the payload' and using the nano 'computer bits' as biosensors in the body. 'It actually has the ability to accumulate data from your body, like your breathing, your respiration, thoughts, emotions, all kinds of things.'

She said the technology obviously has the ability to operate through Wi-Fi and transmit and receive energy, messages, frequencies or impulses. 'Just imagine you're getting this new substance in you and it can react to things all around you, the 5G, your smart device, your phones.' We had something completely foreign in the human body that had never been launched large scale at a time when we were seeing 5G going into schools and hospitals (plus the Musk satellites) and she believed the 'vaccine' transmission had something to do with this: '… if these people have this inside of them … it can act like an antenna and actually transmit it outwardly as well.' The synthetic substance produced its own voltage and so it could have that kind of effect. This fits with my own contention that the nano receiver-transmitters are designed to connect people to the

Smart Grid and break the receiver-transmitter connection to expanded consciousness. That would explain the French energy healer's experience of the disconnection of body from 'soul' with those who have had the 'vaccine'. The nanobots, self-replicating inside the body, would also transmit the synthetic frequency which could be picked up through close interaction by those who have not been 'vaccinated'. Madej speculated that perhaps it was 5G and increased levels of other radiation that was causing the symptoms directly although interestingly she said that non-'vaccinated' patients had shown improvement when they were away from the 'vaccinated' person they had interacted with. It must be remembered that you can control frequency and energy with your mind and you can consciously create energetic barriers or bubbles with the mind to stop damaging frequencies from penetrating your field. American paediatrician Dr Larry Palevsky said the 'vaccine' was not a 'vaccine' and was never designed to protect from a 'viral' infection. He called it 'a massive, brilliant propaganda of genocide' because they didn't have to inject everyone to get the result they wanted. He said the content of the jabs was able to infuse any material into the brain, heart, lungs, kidneys, liver, sperm and female productive system. 'This is genocide; this is a weapon of mass destruction.' At the same time American colleges were banning students from attending if they didn't have this life-changing and potentially life-ending 'vaccine'. Class action lawsuits must follow when the consequences of this college fascism come to light. As the book was going to press came reports about fertility effects on sperm in 'vaccinated' men which would absolutely fit with what I have been saying and hospitals continued to fill with 'vaccine' reactions. Another question is what about transmission via blood transfusions? The NHS has extended blood donation restrictions from seven days after a 'Covid vaccination' to 28 days after even a sore arm reaction.

I said in the spring of 2020 that the then touted 'Covid vaccine' would be ongoing each year like the flu jab. A year later Pfizer CEO, the appalling Albert Bourla, said people would 'likely' need a 'booster dose' of the 'vaccine' within 12 months of getting 'fully

vaccinated' and then a yearly shot. 'Variants will play a key role', he said confirming the point. Johnson & Johnson CEO Alex Gorsky also took time out from his 'vaccine' disaster to say that people may need to be vaccinated against 'Covid-19' each year. UK Health Secretary, the psychopath Matt Hancock, said additional 'boosters' would be available in the autumn of 2021. This is the trap of the 'vaccine passport'. The public will have to accept every last 'vaccine' they introduce, including for the fake 'variants', or it would cease to be valid. The only other way in some cases would be continuous testing with a test not testing for the 'virus' and what is on the swabs constantly pushed up your noise towards the brain every time?

'Vaccines' changing behaviour

I mentioned in the body of the book how I believed we would see gathering behaviour changes in the 'vaccinated' and I am already hearing such comments from the non-'vaccinated' describing behaviour changes in friends, loved ones and work colleagues. This will only increase as the self-replicating synthetic material and nanoparticles expand in body and brain. An article in the *Guardian* in 2016 detailed research at the University of Virginia in Charlottesville which developed a new method for controlling brain circuits associated with complex animal behaviour. The method, dubbed 'magnetogenetics', involves genetically-engineering a protein called ferritin, which stores and releases iron, to create a magnetised substance – 'Magneto' – that can activate specific groups of nerve cells from a distance. This is claimed to be an advance on other methods of brain activity manipulation known as optogenetics and chemogenetics (the Cult has been developing methods of brain control for a long time). The ferritin technique is said to be non-invasive and able to activate neurons 'rapidly and reversibly'. In other words, human thought and perception. The article said that earlier studies revealed how nerve cell proteins 'activated by heat and mechanical pressure can be genetically engineered so that they become sensitive to radio waves and magnetic fields, by attaching them to an iron-storing protein called ferritin, or to inorganic

paramagnetic particles'. Sensitive to radio waves and magnetic fields? You mean like 5G, 6G and 7G? This is the human-AI Smart Grid hive mind we are talking about. The *Guardian* article said:

> … the researchers injected Magneto into the striatum of freely behaving mice, a deep brain structure containing dopamine-producing neurons that are involved in reward and motivation, and then placed the animals into an apparatus split into magnetised and non-magnetised sections.
>
> Mice expressing Magneto spent far more time in the magnetised areas than mice that did not, because activation of the protein caused the striatal neurons expressing it to release dopamine, so that the mice found being in those areas rewarding. This shows that Magneto can remotely control the firing of neurons deep within the brain, and also control complex behaviours.

Make no mistake this basic methodology will be part of the 'Covid vaccine' cocktail and using magnetics to change brain function through electromagnetic field frequency activation. The Pentagon is developing a 'Covid vaccine' using ferritin. Magnetics would explain changes in behaviour and why videos are appearing across the Internet as I write showing how magnets stick to the skin at the point of the 'vaccine' shot. Once people take these 'vaccines' anything becomes possible in terms of brain function and illness which will be blamed on 'Covid-19' and 'variants'. Magnetic field manipulation would further explain why the non-'vaccinated' are reporting the same symptoms as the 'vaccinated' they interact with and why those symptoms are reported to decrease when not in their company. Interestingly 'Magneto', a 'mutant', is a character in the Marvel Comic *X-Men* stories with the ability to manipulate magnetic fields and he believes that mutants should fight back against their human oppressors by any means necessary. The character was born Erik Lehnsherr to a Jewish family in Germany.

Cult-controlled courts

The European Court of Human Rights opened the door for mandatory 'Covid-19 vaccines' across the continent when it ruled in a Czech Republic dispute over childhood immunisation that legally

enforced vaccination could be 'necessary in a democratic society'. The 17 judges decided that compulsory vaccinations did not breach human rights law. On the face of it the judgement was so inverted you gasp for air. If not having a vaccine infused into your body is not a human right then what is? Ah, but they said human rights law which has been specifically written to delete all human rights at the behest of the state (the Cult). Article 8 of the European Convention on Human Rights relates to the right to a private life. The crucial word here is *'except'*:

> There shall be no interference by a public authority with the exercise of this right EXCEPT such as is in accordance with the law and is necessary in a democratic society in the interests of national security, public safety or the economic wellbeing of the country, for the prevention of disorder or crime, for the protection of health or morals, or for the protection of the rights and freedoms of others [My emphasis].

No interference *except* in accordance with the law means there *are* no 'human rights' *except* what EU governments decide you can have at their behest. 'As is necessary in a democratic society' explains that reference in the judgement and 'in the interests of national security, public safety or the economic well-being of the country, for the prevention of disorder or crime, for the protection of health or morals, or for the protection of the rights and freedoms of others' gives the EU a coach and horses to ride through 'human rights' and scatter them in all directions. The judiciary is not a check and balance on government extremism; it is a vehicle to enforce it. This judgement was almost laughably predictable when the last thing the Cult wanted was a decision that went against mandatory vaccination. Judges rule over and over again to benefit the system of which they are a part. Vaccination disputes that come before them are invariably delivered in favour of doctors and authorities representing the view of the state which owns the judiciary. Oh, yes, and we have even had calls to stop putting 'Covid-19' on death certificates within 28 days of a 'positive test' because it is claimed the practice makes the 'vaccine' appear not to work. They are laughing at you.

The scale of madness, inhumanity and things to come was highlighted when those not 'vaccinated' for 'Covid' were refused evacuation from the Caribbean island of St Vincent during massive volcanic eruptions. Cruise ships taking residents to the safety of another island allowed only the 'vaccinated' to board and the rest were left to their fate. Even in life and death situations like this we see 'Covid' stripping people of their most basic human instincts and the insanity is even more extreme when you think that fake 'vaccine'-makers are not even claiming their body-manipulating concoctions stop 'infection' and 'transmission' of a 'virus' that doesn't exist. St Vincent Prime Minister Ralph Gonsalves said: 'The chief medical officer will be identifying the persons already vaccinated so that we can get them on the ship.' Note again the power of the chief medical officer who, like Whitty in the UK, will be answering to the World Health Organization. This is the Cult network structure that has overridden politicians who 'follow the science' which means doing what WHO-controlled 'medical officers' and 'science advisers' tell them. Gonsalves even said that residents who were 'vaccinated' after the order so they could board the ships would still be refused entry due to possible side effects such as 'wooziness in the head'. The good news is that if they were woozy enough in the head they could qualify to be prime minister of St Vincent.

Microchipping freedom

The European judgement will be used at some point to justify moves to enforce the 'Covid' DNA-manipulating procedure. Sandra Ro, CEO of the Global Blockchain Business Council, told a World Economic Forum event that she hoped 'vaccine passports' would help to 'drive forced consent and standardisation' of global digital identity schemes: 'I'm hoping with the desire and global demand for some sort of vaccine passport – so that people can get travelling and working again – [it] will drive forced consent, standardisation, and frankly, cooperation across the world.' The lady is either not very bright, or thoroughly mendacious, to use the term 'forced consent'.

You do not 'consent' if you are forced – you *submit*. She was describing what the plan has been all along and that's to enforce a digital identity on every human without which they could not function. 'Vaccine passports' are opening the door and are far from the end goal. A digital identity would allow you to be tracked in everything you do in cyberspace and this is the same technique used by Cult-owned China to enforce its social credit system of total control. The ultimate 'passport' is planned to be a microchip as my books have warned for nearly 30 years. Those nice people at the Pentagon working for the Cult-controlled Defense Advanced Research Projects Agency (DARPA) claimed in April, 2021, they have developed a microchip inserted under the skin to detect 'asymptomatic Covid-19 infection' before it becomes an outbreak and a 'revolutionary filter' that can remove the 'virus' from the blood when attached to a dialysis machine. The only problems with this are that the 'virus' does not exist and people transmitting the 'virus' with no symptoms is brain-numbing bullshit. This is, of course, not a ruse to get people to be microchipped for very different reasons. DARPA also said it was producing a one-stop 'vaccine' for the 'virus' and all 'variants'. One of the most sinister organisations on Planet Earth is doing this? Better have it then. These people are insane because Wetiko that possesses them is insane.

Researchers from the Salk Institute in California announced they have created an embryo that is part human and part monkey. My books going back to the 1990s have exposed experiments in top secret underground facilities in the United States where humans are being crossed with animal and non-human 'extraterrestrial' species. They are now easing that long-developed capability into the public arena and there is much more to come given we are dealing with psychiatric basket cases. Talking of which – Elon Musk's scientists at Neuralink trained a monkey to play Pong and other puzzles on a computer screen using a joystick and when the monkey made the correct move a metal tube squirted banana smoothie into his mouth which is the basic technique for training humans into unquestioning compliance. Two Neuralink chips were in the monkey's skull and

more than 2,000 wires 'fanned out' into its brain. Eventually the monkey played a video game purely with its brain waves. Psychopathic narcissist Musk said the 'breakthrough' was a step towards putting Neuralink chips into human skulls and merging minds with artificial intelligence. *Exactly*. This man is so dark and Cult to his DNA.

World Economic Fascism (WEF)

The World Economic Forum is telling you the plan by the statements made at its many and various events. Cult-owned fascist YouTube CEO Susan Wojcicki spoke at the 2021 WEF Global Technology Governance Summit (see the name) in which 40 governments and 150 companies met to ensure 'the responsible design and deployment of emerging technologies'. Orwellian translation: 'Ensuring the design and deployment of long-planned technologies will advance the Cult agenda for control and censorship.' Freedom-destroyer and Nuremberg-bound Wojcicki expressed support for tech platforms like hers to censor content that is 'technically legal but could be harmful'. Who decides what is 'harmful'? She does and they do. 'Harmful' will be whatever the Cult doesn't want people to see and we have legislation proposed by the UK government that would censor content on the basis of 'harm' no matter if the information is fair, legal and provably true. Make that *especially* if it is fair, legal and provably true. Wojcicki called for a global coalition to be formed to enforce content moderation standards through automated censorship. This is a woman and mega-censor so self-deluded that she shamelessly accepted a 'free expression' award – *Wojcicki* – in an event sponsored by her own *YouTube*. They have no shame and no self-awareness.

You know that 'Covid' is a scam and Wojcicki a Cult operative when YouTube is censoring medical and scientific opinion purely on the grounds of whether it supports or opposes the Cult 'Covid' narrative. Florida governor Ron DeSantis compiled an expert panel with four professors of medicine from Harvard, Oxford, and Stanford Universities who spoke against forcing children and

vaccinated people to wear masks. They also said there was no proof that lockdowns reduced spread or death rates of 'Covid-19'. Cult-gofer Wojcicki and her YouTube deleted the panel video 'because it included content that contradicts the consensus of local and global health authorities regarding the efficacy of masks to prevent the spread of Covid-19'. This 'consensus' refers to what the Cult tells the World Health Organization to say and the WHO tells 'local health authorities' to do. Wojcicki knows this, of course. The panellists pointed out that censorship of scientific debate was responsible for deaths from many causes, but Wojcicki couldn't care less. She would not dare go against what she is told and as a disgrace to humanity she wouldn't want to anyway. The UK government is seeking to pass a fascist 'Online Safety Bill' to specifically target with massive fines and other means non-censored video and social media platforms to make them censor 'lawful but harmful' content like the Cult-owned Facebook, Twitter, Google and YouTube. What is 'lawful but harmful' would be decided by the fascist Blair-created Ofcom.

Another WEF obsession is a cyber-attack on the financial system and this is clearly what the Cult has planned to take down the bank accounts of everyone – except theirs. Those that think they have enough money for the Cult agenda not to matter to them have got a big lesson coming if they continue to ignore what is staring them in the face. The World Economic Forum, funded by Gates and fronted by Klaus Schwab, announced it would be running a 'simulation' with the Russian government and global banks of just such an attack called Cyber Polygon 2021. What they simulate – as with the 'Covid' Event 201 – they plan to instigate. The WEF is involved in a project with the Cult-owned Carnegie Endowment for International Peace called the WEF-Carnegie Cyber Policy Initiative which seeks to merge Wall Street banks, 'regulators' (I love it) and intelligence agencies to 'prevent' (arrange and allow) a cyber-attack that would bring down the global financial system as long planned by those that control the WEF and the Carnegie operation. The Carnegie Endowment for International Peace sent an instruction to First World

War US President Woodrow Wilson not to let the war end before society had been irreversibly transformed.

The Wuhan lab diversion

As I close, the Cult-controlled authorities and lapdog media are systematically pushing 'the virus was released from the Wuhan lab' narrative. There are two versions – it happened by accident and it happened on purpose. Both are nonsense. The perceived existence of the never-shown-to-exist 'virus' is vital to sell the impression that there is actually an infective agent to deal with and to allow the endless potential for terrifying the population with 'variants' of a 'virus' that does not exist. The authorities at the time of writing are going with the 'by accident' while the alternative media is promoting the 'on purpose'. Cable news host Tucker Carlson who has questioned aspects of lockdown and 'vaccine' compulsion has bought the Wuhan lab story. 'Everyone now agrees' he said. Well, I don't and many others don't and the question is *why* does the system and its media suddenly 'agree'? When the media moves as one unit with a narrative it is always a lie – witness the hour by hour mendacity of the 'Covid' era. Why would this Cult-owned combination which has unleashed lies like machine gun fire suddenly 'agree' to tell the truth??

Much of the alternative media is buying the lie because it fits the conspiracy narrative, but it's the *wrong* conspiracy. The real conspiracy is that *there is no virus* and that is what the Cult is desperate to hide. The idea that the 'virus' was released by accident is ludicrous when the whole 'Covid' hoax was clearly long-planned and waiting to be played out as it was so fast in accordance with the Rockefeller document and Event 201. So they prepared everything in detail over decades and then sat around strumming their fingers waiting for an 'accidental' release from a bio-lab? *What??* It's crazy. Then there's the 'on purpose' claim. You want to circulate a 'deadly virus' and hide the fact that you've done so and you release it down the street from the highest-level bio-lab in China? I repeat – *What??*

You would release it far from that lab to stop any association being made. But, no, we'll do it in a place where the connection was certain to be made. Why would you need to scam 'cases' and 'deaths' and pay hospitals to diagnose 'Covid-19' if you had a real 'virus'? What are sections of the alternative media doing believing this crap? Where were all the mass deaths in Wuhan from a 'deadly pathogen' when the recovery to normal life after the initial propaganda was dramatic in speed? Why isn't the 'deadly pathogen' now circulating all over China with bodies in the street? Once again we have the technique of tell them what they want to hear and they will likely believe it. The alternative media has its 'conspiracy' and with Carlson it fits with his 'China is the danger' narrative over years. China *is* a danger as a global Cult operations centre, but not for this reason. The Wuhan lab story also has the potential to instigate conflict with China when at some stage the plan is to trigger a Problem-Reaction-Solution confrontation with the West. Question everything – *everything* – and especially when the media agrees on a common party line.

Third wave ... fourth wave ... fifth wave ...

As the book went into production the world was being set up for more lockdowns and a 'third wave' supported by invented 'variants' that were increasing all the time and will continue to do so in public statements and computer programs, but not in reality. India became the new Italy in the 'Covid' propaganda campaign and we were told to be frightened of the new 'Indian strain'. Somehow I couldn't find it within myself to do so. A document produced for the UK government entitled 'Summary of further modelling of easing of restrictions – Roadmap Step 2' declared that a third wave was inevitable (of course when it's in the script) and it would be the fault of children and those who refuse the health-destroying fake 'Covid vaccine'. One of the computer models involved came from the Cult-owned *Imperial College* and the other from Warwick University which I wouldn't trust to tell me the date in a calendar factory. The document states that both models presumed extremely high uptake

of the 'Covid vaccines' and didn't allow for 'variants'. The document states: 'The resurgence is a result of some people (mostly children) being ineligible for vaccination; others choosing not to receive the vaccine; and others being vaccinated but not perfectly protected.' The mendacity takes the breath away. Okay, blame those with a brain who won't take the DNA-modifying shots and put more pressure on children to have it as 'trials' were underway involving children as young as six months with parents who give insanity a bad name. Massive pressure is being put on the young to have the fake 'vaccine' and child age consent limits have been systematically lowered around the world to stop parents intervening. Most extraordinary about the document was its claim that the 'third wave' would be driven by 'the resurgence in both hospitalisations and deaths … dominated by *those that have received two doses of the vaccine*, comprising around 60-70% of the wave respectively'. The predicted peak of the 'third wave' suggested 300 deaths per day with 250 of them *fully 'vaccinated' people*. How many more lies do acquiescers need to be told before they see the obvious? Those who took the jab to 'protect themselves' are projected to be those who mostly get sick and die? So what's in the 'vaccine'? The document went on:

> It is possible that a summer of low prevalence could be followed by substantial increases in incidence over the following autumn and winter. Low prevalence in late summer should not be taken as an indication that SARS-CoV-2 has retreated or that the population has high enough levels of immunity to prevent another wave.

They are telling you the script and while many British people believed 'Covid' restrictions would end in the summer of 2021 the government was preparing for them to be ongoing. Authorities were awarding contracts for 'Covid marshals' to police the restrictions with contracts starting in July, 2021, and going through to January 31st, 2022, and the government was advertising for 'Media Buying Services' to secure media propaganda slots worth a potential £320 million for 'Covid-19 campaigns' with a contract not ending until March, 2022. The recipient – via a list of other front companies – was reported to be American media marketing giant Omnicom Group

Inc. While money is no object for 'Covid' the UK waiting list for all other treatment – including life-threatening conditions – passed 4.5 million. Meantime the Cult is seeking to control all official 'inquiries' to block revelations about what has really been happening and why. It must not be allowed to – we need Nuremberg jury trials in every country. The cover-up doesn't get more obvious than appointing ultra-Zionist professor Philip Zelikow to oversee two dozen US virologists, public health officials, clinicians, former government officials and four American 'charitable foundations' to 'learn the lessons' of the 'Covid' debacle. The personnel will be those that created and perpetuated the 'Covid' lies while Zelikow is the former executive director of the 9/11 Commission who ensured that the truth about those attacks never came out and produced a report that must be among the most mendacious and manipulative documents ever written – see *The Trigger* for the detailed exposure of the almost unimaginable 9/11 story in which Sabbatians can be found at every level.

Passive no more

People are increasingly challenging the authorities with amazing numbers of people taking to the streets in London well beyond the ability of the Face-Nappies to stop them. Instead the Nappies choose situations away from the mass crowds to target, intimidate, and seek to promote the impression of 'violent protestors'. One such incident happened in London's Hyde Park. Hundreds of thousands walking through the streets in protest against 'Covid' fascism were ignored by the Cult-owned BBC and most of the rest of the mainstream media, but they delighted in reporting how police were injured in 'clashes with protestors'. The truth was that a group of people gathered in Hyde Park at the end of one march when most had gone home and they were peacefully having a good time with music and chat. Face-Nappies who couldn't deal with the full-march crowd then waded in with their batons and got more than they bargained for. Instead of just standing for this criminal brutality the crowd used their numerical superiority to push the Face-Nappies out of the

park. Eventually the Nappies turned and ran. Unfortunately two or three idiots in the crowd threw drink cans striking two officers which gave the media and the government the image they wanted to discredit the 99.9999 percent who were peaceful. The idiots walked straight into the trap and we must always be aware of potential agent provocateurs used by the authorities to discredit their targets.

This response from the crowd – the can people apart – must be a turning point when the public no longer stand by while the innocent are arrested and brutally attacked by the Face-Nappies. That doesn't mean to be violent, that's the last thing we need. We'll leave the violence to the Face-Nappies and government. But it does mean that when the Face-Nappies use violence against peaceful people the numerical superiority is employed to stop them and make citizen's arrests or Common Law arrests for a breach of the peace. The time for being passive in the face of fascism is over.

We are the many, they are the few, and we need to make that count before there is no freedom left and our children and grandchildren face an ongoing fascist nightmare.

COME ON PEOPLE – IT'S TIME.

One final thought …

The power of love
A force from above
Cleaning my soul
Flame on burn desire
Love with tongues of fire
Purge the soul
Make love your goal

I'll protect you from the hooded claw
Keep the vampires from your door
When the chips are down I'll be around
With my undying, death-defying
Love for you

Envy will hurt itself
Let yourself be beautiful
Sparkling love, flowers
And pearls and pretty girls
Love is like an energy
Rushin' rushin' inside of me

This time we go sublime
Lovers entwine, divine, divine,
Love is danger, love is pleasure
Love is pure – the only treasure

I'm so in love with you
Purge the soul
Make love your goal

The power of love
A force from above
Cleaning my soul
The power of love
A force from above
A sky-scraping dove

Flame on burn desire
Love with tongues of fire
Purge the soul
Make love your goal

Frankie Goes To Hollywood

APPENDIX

Cowan-Kaufman-Morell Statement on Virus Isolation (SOVI)

Isolation: The action of isolating; the fact or condition of being isolated or standing alone; separation from other things or persons; solitariness
Oxford English Dictionary

The controversy over whether the SARS-CoV-2 virus has ever been isolated or purified continues. However, using the above definition, common sense, the laws of logic and the dictates of science, any unbiased person must come to the conclusion that the SARS-CoV-2 virus has never been isolated or purified. As a result, no confirmation of the virus' existence can be found. The logical, common sense, and scientific consequences of this fact are:

- the structure and composition of something not shown to exist can't be known, including the presence, structure, and function of any hypothetical spike or other proteins;
- the genetic sequence of something that has never been found can't be known;
- "variants" of something that hasn't been shown to exist can't be known;
- it's impossible to demonstrate that SARS-CoV-2 causes a disease called Covid-19.

In as concise terms as possible, here's the proper way to isolate, characterize and demonstrate a new virus. First, one takes samples (blood, sputum, secretions) from many people (e.g. 500) with symptoms which are unique and specific enough to characterize an illness. Without mixing these samples with ANY tissue or products that also contain genetic material, the virologist macerates, filters and ultracentrifuges i.e. *purifies* the specimen. This common virology technique, done for decades to isolate bacteriophages[1] and so-called giant viruses in every virology lab, then allows the virologist to demonstrate with electron microscopy thousands of identically sized and shaped particles. These particles are the isolated and purified virus.

These identical particles are then checked for uniformity by physical and/or microscopic techniques. Once the purity is determined, the particles may be further characterized. This would include examining the structure, morphology, and chemical composition of the particles. Next, their genetic makeup is characterized by extracting the genetic material directly from the purified particles and using genetic-sequencing techniques, such as Sanger sequencing, that have also been around for decades. Then one does an analysis to confirm that these uniform particles are exogenous (outside) in origin as a virus is conceptualized to be, and not the normal breakdown products of dead and dying tissues.[2] (As of May 2020, we know that virologists have no way to determine whether the particles they're seeing are viruses or just normal breakdown products of dead and dying tissues.)[3]

---

1 Isolation, characterization and analysis of bacteriophages from the haloalkaline lake Elmenteita, KenyaJuliah Khayeli Akhwale et al, PLOS One, Published: April 25, 2019. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0215734 – accessed 2/15/21

2 "Extracellular Vesicles Derived From Apoptotic Cells: An Essential Link Between Death and Regeneration," Maojiao Li1 et al, Frontiers in Cell and Developmental Biology, 2020 October 2. https://www.frontiersin.org/articles/10.3389/fcell.2020.573511/full – accessed 2/15/21

3 "The Role of Extraellular Vesicles as Allies of HIV, HCV and SARS Viruses," Flavia Giannessi, et al, Viruses, 2020 May

If we have come this far then we have fully isolated, characterized, and genetically sequenced an exogenous virus particle. However, we still have to show it is causally related to a disease. This is carried out by exposing a group of healthy subjects (animals are usually used) to this isolated, purified virus in the manner in which the disease is thought to be transmitted. If the animals get sick with the same disease, as confirmed by clinical and autopsy findings, one has now shown that the virus actually causes a disease. This demonstrates infectivity and transmission of an infectious agent.

None of these steps has even been attempted with the SARS-CoV-2 virus, nor have all these steps been successfully performed for any so-called pathogenic virus. Our research indicates that a single study showing these steps does not exist in the medical literature.

Instead, since 1954, virologists have taken unpurified samples from a relatively few people, often less than ten, with a similar disease. They then minimally process this sample and inoculate this unpurified sample onto tissue culture containing usually four to six other types of material – all of which contain identical genetic material as to what is called a "virus." The tissue culture is starved and poisoned and naturally disintegrates into many types of particles, some of which contain genetic material. Against all common sense, logic, use of the English language and scientific integrity, this process is called "virus isolation." This brew containing fragments of genetic material from many sources is then subjected to genetic analysis, which then creates in a computer-simulation process the alleged sequence of the alleged virus, a so called in silico genome. At no time is an actual virus confirmed by electron microscopy. At no time is a genome extracted and sequenced from an actual virus. This is scientific fraud.

The observation that the unpurified specimen — inoculated onto tissue culture along with toxic antibiotics, bovine fetal tissue, amniotic fluid and other tissues — destroys the kidney tissue onto which it is inoculated is given as evidence of the virus' existence and pathogenicity. This is scientific fraud.

From now on, when anyone gives you a paper that suggests the SARS-CoV-2 virus has been isolated, please check the methods sections. If the researchers used Vero cells or any other culture method, you know that their process was not isolation. You will hear the following excuses for why actual isolation isn't done:

1. There were not enough virus particles found in samples from patients to analyze.

2. Viruses are intracellular parasites; they can't be found outside the cell in this manner.

If No. 1 is correct, and we can't find the virus in the sputum of sick people, then on what evidence do we think the virus is dangerous or even lethal? If No. 2 is correct, then how is the virus spread from person to person? We are told it emerges from the cell to infect others. Then why isn't it possible to find it?

Finally, questioning these virology techniques and conclusions is not some distraction or divisive issue. Shining the light on this truth is essential to stop this terrible fraud that humanity is confronting. For, as we now know, if the virus has never been isolated, sequenced or shown to cause illness, if the virus is imaginary, then why are we wearing masks, social distancing and putting the whole world into prison?

Finally, if pathogenic viruses don't exist, then what is going into those injectable devices erroneously called "vaccines," and what is their purpose? This scientific question is the most urgent and relevant one of our time.

We are correct. The SARS-CoV2 virus does not exist.

Sally Fallon Morell, MA
Dr. Thomas Cowan, MD
Dr. Andrew Kaufman, MD

Bibliography

Alinsky, Saul: *Rules for Radicals* (Vintage, 1989)

Antelman, Rabbi Marvin: *To Eliminate the Opiate* (Zahavia, 1974)

Bastardi, Joe: *The Climate Chronicles* (Relentless Thunder Press, 2018)

Cowan, Tom: *Human Heart, Cosmic Heart* (Chelsea Green Publishing, 2016)

Cowan, Tom, and Fallon Morell, Sally: *The Contagion Myth* (Skyhorse Publishing, 2020)

Forbes, Jack D: *Columbus And Other Cannibals – The Wetiko Disease of Exploitation, Imperialism, and Terrorism* (Seven Stories Press, 2008 – originally published in 1979)

Gates, Bill: *How to Avoid a Climate Disaster: The Solutions We Have and the Breakthroughs We Need* (Allen Lane, 2021)

Huxley, Aldous: *Brave New World* (Chatto & Windus, 1932)

Köhnlein, Dr Claus, and Engelbrecht, Torsten: *Virus Mania* (emu-Vertag, Lahnstein, 2020)

Lanza, Robert, and Berman, Bob: *Biocentrism* (BenBella Books, 2010)

Lash, John Lamb: *Not In His Image* (Chelsea Green Publishing, 2006)

Lester, Dawn, and Parker, David: *What Really Makes You Ill – Why everything you thought you knew about disease is wrong* (Independently Published, 2019)

Levy, Paul: *Dispelling Wetiko, Breaking the Spell of Evil* (North Atlantic Books, 2013)

Marx, Karl: *A World Without Jews* (Philosophical Library, first edition, 1959)

Mullis, Kary: *Dancing Naked in the Mine Field* (Bloomsbury, 1999)

O'Brien, Cathy: *Trance-Formation of America* (Reality Marketing, 1995)

Scholem, Gershon: *The Messianic Idea in Judaism* (Schocken Books, 1994)

Schwab, Klaus, and Davis, Nicholas: *Shaping the Future of the Fourth Industrial Revolution: A guide to building a better world* (Penguin Books, 2018)

Schwab, Klaus: *The Great Reset* (Agentur Schweiz, 2020)

Sunstein, Cass and Thaler, Richard: *Nudge: Improving Decisions About Health, Wealth, and Happiness* (Penguin, 2009)

Swan, Shanna: *Count Down: How Our Modern World Is Threatening Sperm Counts, Altering Male and Female Reproductive Development and Imperiling the Future of the Human Race* (Scribner, 2021)

Tegmark, Max: *Our Mathematical Universe: My Quest for the Ultimate Nature of Reality* (Penguin, 2015)

Velikovsky, Immanuel: *Worlds in Collision* (Paradigma, 2009)

Wilton, Robert: *The Last Days of the Romanovs* (Blurb, 2018, first published 1920)

Index

A

abusive relationships
blaming themselves, abused as ref1
children ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10
conspiracy theories ref1
domestic abuse ref1, ref2
economic abuse and dependency ref1
isolation ref1
physical abuse ref1
psychological abuse ref1
signs of abuse ref1
addiction
alcoholism ref1
frequencies ref1
substance abuse ref1, ref2
technology ref1, ref2, ref3
Adelson, Sheldon ref1, ref2, ref3
Agenda 21/Agenda 2030 (UN) ref1, ref2, ref3, ref4
AIDs/HIV ref1
causal link between HIV and AIDs ref1, ref2
retroviruses ref1
testing ref1, ref2
trial-run for Covid-19, as ref1, ref2
aliens/extraterrestrials ref1, ref2
aluminium ref1
Amazon ref1, ref2, ref3

amplification cycles ref1, ref2
anaphylactic shock ref1, ref2, ref3, ref4
animals ref1, ref2, ref3
antibodies ref1, ref2, ref3, ref4, ref5
Antifa ref1, ref2, ref3, ref4
antigens ref1, ref2
anti-Semitism ref1, ref2, ref3
Archons ref1, ref2
  consciousness ref1, ref2, ref3
  energy ref1, ref2, ref3
  ennoia ref1
  genetic manipulation ref1, ref2
  inversion ref1, ref2, ref3
  lockdowns ref1
  money ref1
  radiation ref1
  religion ref1, ref2
  technology ref1, ref2, ref3
  Wetiko factor ref1, ref2, ref3, ref4
artificial intelligence (AI) ref1
army made up of robots ref1, ref2
  Human 2.0 ref1, ref2
  Internet ref1
  MHRA ref1
  Morgellons fibres ref1, ref2
  Smart Grid ref1
  Wetiko factor ref1
asymptomatic, Covid-19 as ref1, ref2, ref3
aviation industry ref1

B

banking, finance and money ref1, ref2, ref3
2008 crisis ref1, ref2
boom and bust ref1
cashless digital money systems ref1
central banks ref1
credit ref1
digital currency ref1
fractional reserve lending ref1
Great Reset ref1
guaranteed income ref1, ref2, ref3
Human 2.0 ref1
incomes, destruction of ref1, ref2
interest ref1
one per cent ref1, ref2
scams ref1
BBC ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8
Becker-Phelps, Leslie ref1
Behavioural Insights Team (BIT) (Nudge Unit) ref1, ref2, ref3
behavioural scientists *and* **psychologists, advice from** ref1, ref2
Bezos, Jeff ref1, ref2, ref3, ref4
Biden, Hunter ref1
Biden, Joe ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10, ref11, ref12, ref13, ref14, ref15, ref16, ref17
Big Pharma
cholesterol ref1
health professionals ref1, ref2
immunity from prosecution in US ref1
vaccines ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8
Wetiko factor ref1, ref2
WHO ref1, ref2, ref3
Bill and Melinda Gates Foundation ref1, ref2, ref3, ref4, ref5, ref6, ref7

billionaires ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9 ref10, ref11
bird flu (H5N1) ref1
Black Lives Matter (BLM) ref1, ref2, ref3, ref4, ref5
Blair, Tony ref1, ref2, ref3, ref4, ref5, ref6, ref7
Brin, Sergei ref1, ref2, ref3, ref4, ref5, ref6, ref7
British Empire ref1
Bush, George HW ref1, ref2
Bush, George W ref1, ref2, ref3, ref4
Byrd, Robert ref1

C

Canada
  Global Cult ref1
  hate speech ref1
  internment ref1
  masks ref1
  old people ref1
  SARS-COV-2 ref1
  satellites ref1
  vaccines ref1
  wearable technology ref1
Capitol Hill riot ref1, ref2
  agents provocateur ref1
  Antifa ref1
  Black Lives Matter (BLM) ref1, ref2
  QAnon ref1
  security precautions, lack of ref1, ref2, ref3
carbon dioxide ref1, ref2
care homes, deaths in ref1, ref2
cashless digital money systems ref1
censorship ref1, ref2, ref3, ref4, ref5

fact-checkers ref1
masks ref1
media ref1, ref2
private messages ref1
social media ref1, ref2, ref3, ref4, ref5, ref6
transgender persons ref1
vaccines ref1, ref2, ref3
Wokeness ref1
Centers for Disease Control (CDC) (United States) ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10, ref11, ref12, ref13
centralisation ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8
chakras ref1
change agents ref1, ref2, ref3
chemtrails ref1, ref2, ref3
chief medical officers and scientific advisers ref1, ref2, ref3, ref4, ref5, ref6
children *see also* **young people**
abuse ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10
care, taken into ref1, ref2, ref3
education ref1, ref2, ref3, ref4
energy ref1
family courts ref1
hand sanitisers ref1
human sacrifice ref1
lockdowns ref1, ref2, ref3
masks ref1, ref2, ref3, ref4, ref5
mental health ref1
old people ref1
parents, replacement of ref1, ref2
Psyop (psychological operation), Covid as a ref1, ref2
reframing ref1
smartphone addiction ref1

social distancing and isolation ref1
social media ref1
transgender persons ref1, ref2
United States ref1
vaccines ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10
Wetiko factor ref1
China ref1, ref2, ref3, ref4
anal swab tests ref1
Chinese Revolution ref1, ref2, ref3
digital currency ref1
Global Cult ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9
guaranteed income ref1
Imperial College ref1
Israel ref1
lockdown ref1, ref2
masculinity crisis ref1
masks ref1
media ref1
origins of virus in China ref1, ref2, ref3, ref4, ref5
pollution causing respiratory diseases ref1
Sabbatians ref1, ref2
Smart Grid ref1, ref2
social credit system ref1
testing ref1, ref2
United States ref1, ref2
vaccines ref1, ref2
Wetiko factor ref1
wet market conspiracy ref1
Wuhan ref1, ref2, ref3, ref4, ref5, ref6, ref7
cholesterol ref1, ref2
Christianity ref1, ref2, ref3, ref4, ref5
criticism ref1
cross, inversion of the ref1

Nag Hammadi texts ref1, ref2, ref3
Roman Catholic Church ref1, ref2
Sabbatians ref1, ref2
Satan ref1, ref2, ref3, ref4
Wokeness ref1
class ref1, ref2
climate change hoax ref1, ref2, ref3, ref4, ref5
Agenda 21/Agenda 2030 ref1, ref2, ref3
carbon dioxide ref1, ref2
Club of Rome ref1, ref2, ref3, ref4, ref5
fear ref1
funding ref1
Global Cult ref1
green new deals ref1
green parties ref1
inversion ref1
perception, control of ref1
PICC ref1
reframing ref1
temperature, increases in ref1
United Nations ref1, ref2
Wikipedia ref1
Wokeness ref1, ref2
Clinton, Bill ref1, ref2, ref3, ref4, ref5, ref6
Clinton, Hillary ref1, ref2, ref3
the cloud ref1, ref2, ref3, ref4, ref5, ref6, ref7
Club of Rome and climate change hoax ref1, ref2, ref3, ref4, ref5
cognitive therapy ref1
Cohn, Roy ref1
Common Law ref1
Admiralty Law ref1
arrests ref1, ref2

contractual law, Statute Law as ref1
corporate entities, people as ref1
legalese ref1
sea, law of the ref1
Statute Law ref1
Common Purpose leadership programme ref1, ref2
communism ref1, ref2
co-morbidities ref1
computer-generated virus,
Covid-19 as ref1, ref2, ref3
computer models ref1, ref2, ref3, ref4, ref5
connections ref1, ref2, ref3, ref4
consciousness ref1, ref2, ref3, ref4
Archons ref1, ref2, ref3
expanded ref1, ref2, ref3, ref4, ref5, ref6, ref7
experience ref1
heart ref1
infinity ref1, ref2
religion ref1, ref2
self-identity ref1
simulation thesis ref1
vaccines ref1
Wetiko factor ref1, ref2
conspiracy theorists ref1, ref2, ref3, ref4, ref5
contradictory rules ref1
contrails ref1
Corman-Drosten test ref1, ref2, ref3, ref4
countermimicry ref1, ref2, ref3
Covid-19 vaccines *see* vaccines
Covidiots ref1, ref2
Cowan, Tom ref1, ref2, ref3, ref4
crimes against humanity ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8

cyber-operations ref1
cyberwarfare ref1

D

DARPA (Defense Advanced Research Projects Agency) ref1
deaths
  care homes ref1
  certificates ref1, ref2, ref3, ref4
  mortality rate ref1
  post-mortems/autopsies ref1
  recording ref1, ref2, ref3, ref4, ref5, ref6, ref7
  vaccines ref1, ref2, ref3, ref4, ref5
deceit
  pyramid of deceit ref1, ref2
  sequence of deceit ref1
decoding ref1, ref2, ref3
dehumanisation ref1, ref2, ref3
Delphi technique ref1
democracy ref1
dependency ref1, ref2, ref3, ref4, ref5
Descartes, René ref1
DNA
  numbers ref1
  vaccines ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10
DNR (do not resuscitate)
  orders ref1
domestic abuse ref1, ref2
downgrading of Covid-19 ref1
Drosten, Christian ref1, ref2, ref3, ref4, ref5, ref6, ref7
Duesberg, Peter ref1, ref2

E

economic abuse ref1
Edmunds, John ref1, ref2
education ref1, ref2, ref3, ref4
electromagnetic spectrum ref1, ref2
Enders, John ref1
energy
  Archons ref1, ref2, ref3
  children and young people ref1
  consciousness ref1
  decoding ref1
  frequencies ref1, ref2, ref3, ref4
  heart ref1
  human energy field ref1
  source, humans as an energy ref1, ref2
  vaccines ref1
  viruses ref1
ennoia ref1
Epstein, Jeffrey ref1, ref2
eternal 'I' ref1, ref2
ethylene oxide ref1
European Union ref1, ref2, ref3, ref4
Event ref1 *and* **Bill Gates** ref2
exosomes, Covid-19 as natural defence mechanism called ref1
experience ref1, ref2
Extinction Rebellion ref1, ref2

F

Facebook
  addiction ref1, 448–50
  Facebook

Archons ref1
censorship ref1, ref2, ref3
hate speech ref1
monopoly, as ref1
private messages, censorship of ref1
Sabbatians ref1
United States election fraud ref1
vaccines ref1
Wetiko factor ref1
fact-checkers ref1
Fauci, Anthony ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10, ref11, ref12
fear ref1, ref2, ref3, ref4
climate change ref1
computer models ref1
conspiracy theories ref1
empty hospitals ref1
Italy ref1, ref2, ref3
lockdowns ref1, ref2, ref3, ref4
masks ref1, ref2
media ref1, ref2
medical staff ref1
Psyop (psychological operation), Covid as a ref1
Wetiko factor ref1, ref2
female infertility ref1
Fermi Paradox ref1
Ferguson, Neil ref1, ref2, ref3, ref4, ref5, ref6, ref7
fertility, decline in ref1
The Field ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8
finance *see* **banking, finance and money**
five-senses ref1, ref2
Archons ref1, ref2, ref3

censorship ref1
consciousness, expansion of ref1, ref2, ref3, ref4, ref5, ref6
decoding ref1
education ref1, ref2
the Field ref1, ref2
God, personification of ref1
infinity ref1, ref2
media ref1
paranormal ref1
perceptual programming ref1, ref2
Phantom Self ref1
pneuma not nous, using ref1
reincarnation ref1
self-identity ref1
Wetiko factor ref1, ref2, ref3, ref4, ref5, ref6
5G ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8
Floyd, George and protests, killing of ref1
flu, re-labelling of ref1, ref2, ref3
food and water, control of ref1, ref2
Freemasons ref1, ref2, ref3, ref4, ref5, ref6
Frei, Rosemary ref1
frequencies
addictions ref1
Archons ref1, ref2, ref3
awareness ref1
chanting and mantras ref1
consciousness ref1
decoding ref1, ref2
education ref1
electromagnetic (EMF) frequencies ref1
energy ref1, ref2, ref3, ref4
fear ref1

the Field ref1, ref2 5G ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10
five-senses ref1, ref2
ghosts ref1
Gnostics ref1
hive-minds ref1
human, meaning of ref1
light ref1, ref2
love ref1, ref2
magnetism ref1
perception ref1
reality ref1, ref2, ref3
simulation ref1
terror ref1
vaccines ref1
Wetiko ref1, ref2, ref3
Fuellmich, Reiner ref1, ref2, ref3
furlough/rescue payments ref1

G

Gallo, Robert ref1, ref2, ref3
Gates, Bill
Archons ref1, ref2, ref3
climate change ref1, ref2, ref3, ref4
Daily Pass tracking system ref1
Epstein ref1
fascism ref1
five senses ref1
GAVI ref1
Great Reset ref1
GSK ref1
Imperial College ref1, ref2
Johns Hopkins University ref1, ref2, ref3

lockdowns ref1, ref2
masks ref1
Nuremberg trial, proposal for ref1, ref2
Rockefellers ref1, ref2
social distancing and isolation ref1
Sun, dimming the ref1
synthetic meat ref1, ref2
vaccines ref1, ref2, ref3, ref4, ref5, ref6, ref7
Wellcome Trust ref1
Wetiko factor ref1, ref2, ref3
WHO ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10
Wokeness ref1
World Economic Forum ref1, ref2, ref3, ref4
Gates, Melinda ref1, ref2, ref3
GAVI vaccine alliance ref1
genetics, manipulation of ref1, ref2, ref3
Germany ref1, ref2, ref3, ref4, ref5, ref6 *see also* **Nazi Germany**
Global Cult ref1, ref2, ref3, ref4, ref5
anti-human, why Global Cult is ref1
Black Lives Matter (BLM) ref1, ref2, ref3, ref4
China ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9
climate change hoax ref1
contradictory rules ref1
Covid-19 ref1, ref2, ref3
fascism ref1
geographical origins ref1
immigration ref1
Internet ref1
mainstream media ref1, ref2
masks ref1, ref2
monarchy ref1
non-human dimension ref1

perception ref1
political parties ref1, ref2
pyramidal hierarchy ref1, ref2, ref3
reframing ref1
Sabbantian-Frankism ref1, ref2
science, manipulation of ref1
spider and the web ref1
transgender persons ref1
vaccines ref1
who controls the Cult ref1
Wokeness ref1, ref2, ref3, ref4
globalisation ref1, ref2
Gnostics ref1, ref2, ref3, ref4, ref5
Google ref1, ref2, ref3, ref4
government
behavioural scientists and psychologists, advice from ref1, ref2
definition ref1
Joint Biosecurity Centre (JBC) ref1
people, abusive relationship with ref1
Great Reset ref1, ref2, ref3, ref4, ref5, ref6
fascism ref1, ref2, ref3
financial system ref1
Human 2.0 ref1
water and food, control of ref1
green parties ref1
Griesz-Brisson, Margarite ref1
guaranteed income ref1, ref2, ref3

H

Hancock, Matt ref1, ref2, ref3, ref4, ref5
hand sanitisers ref1
heart ref1, ref2

hive-minds/groupthink ref1, ref2, ref3
holographs ref1, ref2, ref3, ref4
hospitals, empty ref1
human, meaning of ref1
Human 2.0 ref1
addiction to technology ref1
artificial intelligence (AI) ref1, ref2
elimination of Human 1.0 ref1
fertility, decline in ref1
Great Reset ref1
implantables ref1
money ref1
mRNA ref1
nanotechnology ref1
parents, replacement of ref1, ref2
Smart Grid, connection to ref1, ref2
synthetic biology ref1, ref2, ref3, ref4
testosterone levels, decrease in ref1
transgender = transhumanism ref1, ref2, ref3
vaccines ref1, ref2, ref3, ref4
human sacrifice ref1, ref2, ref3
Hunger Games Society ref1, ref2, ref3, ref4, ref5, ref6, ref7
Huxley, Aldous ref1, ref2, ref3

I

identity politics ref1, ref2, ref3
Illuminati ref1, ref2
illusory physical reality ref1
immigration ref1, ref2, ref3, ref4
Imperial College ref1, ref2, ref3, ref4, ref5, ref6
implantables ref1, ref2

incomes, destruction of ref1, ref2
Infinite Awareness ref1, ref2, ref3, ref4
Internet ref1, ref2 *see also* social media
  artificial intelligence (AI) ref1
  independent journalism, lack of ref1
  Internet of Bodies (IoB) ref1
Internet of Everything (IoE) ref1, ref2
Internet of Things (IoT) ref1, ref2
lockdowns ref1
Psyop (psychological operation), Covid as a ref1
  trolls ref1
intersectionality ref1
inversion
  Archons ref1, ref2, ref3
  climate change hoax ref1
  energy ref1
  Judaism ref1, ref2, ref3
  symbolism ref1
  Wetiko factor ref1
  Wokeness ref1, ref2, ref3
Islam
  Archons ref1
  crypto-Jews ref1
  Islamic State ref1, ref2
  Jinn and Djinn ref1, ref2, ref3
  Ottoman Empire ref1
  Wahhabism ref1
isolation *see* **social distancing** *and* **isolation**
Israel
  China ref1
  Cyber Intelligence Unit Beersheba complex ref1
  expansion of illegal settlements ref1

formation ref1
Global Cult ref1
Judaism ref1, ref2, ref3, ref4, ref5
medical experiments, consent for ref1
Mossad ref1, ref2, ref3, ref4
Palestine-Israel conflict ref1, ref2, ref3
parents, replacement of ref1
Sabbatians ref1, ref2, ref3, ref4, ref5
September 11, 2001, terrorist attacks on United States ref1
Silicon Valley ref1
Smart Grid ref1, ref2
United States ref1, ref2
vaccines ref1
Wetiko factor ref1

Italy
fear ref1, ref2, ref3
Lombardy ref1, ref2, ref3
vaccines ref1

J

Johns Hopkins University ref1, ref2, ref3, ref4, ref5, ref6, ref7
Johnson, Boris ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8
Joint Biosecurity Centre (JBC) ref1
Judaism
anti-Semitism ref1, ref2, ref3
Archons ref1, ref2
crypto-Jews ref1
inversion ref1, ref2, ref3
Israel ref1, ref2, ref3, ref4, ref5
Labour Party ref1
Nazi Germany ref1, ref2, ref3, ref4
Sabbatians ref1, ref2, ref3, ref4, ref5

Silicon Valley ref1
Torah ref1
United States ref1, ref2
Zionists ref1, ref2, ref3

K

Kaufman, Andrew ref1, ref2, ref3, ref4
knowledge ref1, ref2, ref3, ref4, ref5, ref6
Koch's postulates ref1
Kurzweil, Ray ref1, ref2, ref3, ref4, ref5, ref6, ref7
Kushner, Jared **ref1**, **ref2**

L

Labour Party ref1, ref2
Lanka, Stefan ref1, ref2
Lateral Flow Device (LFD) ref1
Levy, Paul ref1, ref2, ref3
Life Program ref1
lockdowns ref1, ref2, ref3
amplification tampering ref1
Archons ref1
Behavioural Insights Team ref1
Black Lives Matter (BLM) ref1
care homes, deaths in ref1
children
abuse ref1, ref2
mental health ref1
China ref1, ref2
computer models ref1
consequences ref1, ref2
dependency ref1, ref2, ref3

domestic abuse ref1
fall in cases ref1
fear ref1, ref2, ref3, ref4
guaranteed income ref1
Hunger Games Society ref1, ref2, ref3
interaction, destroying ref1
Internet ref1, ref2
overdoses ref1
perception ref1
police-military state ref1, ref2
protests ref1, ref2, ref3, ref4, ref5
psychopathic personality ref1, ref2, ref3
reporting/snitching, encouragement of ref1, ref2
testing ref1
vaccines ref1
Wetiko factor ref1
WHO ref1
love ref1, ref2, ref3
Lucifer ref1, ref2, ref3

M

Madej, Carrie ref1, ref2
Magufuli, John ref1, ref2
mainstream media ref1
BBC ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8
censorship ref1, ref2
China ref1
climate change hoax ref1
fear ref1, ref2
Global Cult ref1, ref2
independent journalism, lack of ref1
Ofcom ref1, ref2, ref3

perception ref1, ref2
Psyop (psychological operation), Covid as a ref1
Sabbatians ref1, ref2
social disapproval ref1
social distancing and isolation ref1
United States ref1, ref2
vaccines ref1, ref2, ref3, ref4, ref5
Mao Zedong ref1, ref2, ref3
Marx and Marxism ref1, ref2, ref3, ref4, ref5, ref6
masculinity ref1
masks/face coverings ref1, ref2, ref3
censorship ref1
children ref1, ref2, ref3, ref4, ref5
China, made in ref1
dehumanisation ref1, ref2, ref3
fear ref1, ref2
flu ref1
health professionals ref1, ref2, ref3, ref4
isolation ref1
laughter ref1
mass non-cooperation ref1
microplastics, risk of ref1
mind control ref1
multiple masks ref1
oxygen deficiency ref1, ref2, ref3
police ref1, ref2, ref3, ref4, ref5
pollution, as cause of plastic ref1
Psyop (psychological operation), Covid as a ref1
reframing ref1, ref2
risk assessments, lack of ref1, ref2
self-respect ref1
surgeons ref1

United States ref1
vaccines ref1, ref2, ref3, ref4, ref5
Wetiko factor ref1
'worms' ref1
The Matrix movies ref1, ref2, ref3
measles ref1, ref2
media see mainstream media
Medicines and Healthcare products Regulatory Agency (MHRA) ref1, ref2, ref3, ref4
Mesopotamia ref1
messaging ref1
military-police state ref1, ref2, ref3
mind control ref1, ref2, ref3, ref4, ref5, ref6 *see also* **MKUltra**
MKUltra ref1, ref2, ref3
monarchy ref1
money *see* **banking, finance and money**
Montagnier, Luc ref1, ref2, ref3
Mooney, Bel ref1
Morgellons disease ref1, ref2
mortality rate ref1
Mullis, Kary ref1, ref2, ref3
Musk, Elon ref1

N

Nag Hammadi texts ref1, ref2, ref3
nanotechnology ref1, ref2, ref3
narcissism ref1
Nazi Germany ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8
near-death experiences ref1, ref2
Neocons ref1, ref2, ref3

Neuro-Linguistic Programming (NLP) and the Delphi technique ref1
NHS (National Health Service)
amplification cycles ref1
Common Purpose ref1, ref2
mind control ref1
NHS England ref1
saving the NHS ref1, ref2
vaccines ref1, ref2, ref3, ref4, ref5
whistle-blowers ref1, ref2, ref3
No-Problem-Reaction-Solution ref1, ref2, ref3, ref4
non-human dimension of Global Cult ref1
nous ref1
numbers, reality as ref1
Nuremberg Codes ref1, ref2, ref3
Nuremberg-like tribunal, proposal for ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10, ref11, ref12

O

Obama, Barack ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10
O'Brien, Cathy ref1, ref2, ref3, ref4
Ochel, Evita ref1
Ofcom ref1, ref2, ref3
old people ref1, ref2, ref3, ref4, ref5
Oneness ref1, ref2, ref3
Open Society Foundations (Soros) ref1, ref2, ref3
oxygen 406, 528–34

P

paedophilia ref1, ref2
Page, Larry ref1, ref2, ref3, ref4, ref5, ref6, ref7

Palestine-Israel conflict ref1, ref2, ref3
pandemic, definition of ref1
pandemic and health crisis scenarios/simulations ref1, ref2, ref3, ref4
paranormal ref1
PCR tests ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8
Pearl Harbor attacks, prior knowledge of ref1
Pelosi, Nancy ref1, ref2, ref3
perception ref1, ref2, ref3, ref4
  climate change hoax ref1
  control ref1, ref2, ref3
  decoding ref1, ref2
  enslavement ref1
  externally-delivered perceptions ref1
  five senses ref1
  human labels ref1
  media ref1, ref2
  political parties ref1, ref2
  Psyop (psychological operation), Covid as a ref1
  sale of perception ref1
  self-identity ref1, ref2
  Wokeness ref1
Phantom Self ref1, ref2, ref3
pharmaceutical industry *see* **Big Pharma**
phthalates ref1
Plato's Allegory of the Cave ref1, ref2
pneuma ref1
police
  Black Lives Matter (BLM) ref1
  brutality ref1
  citizen's arrests ref1, ref2
  common law arrests ref1, ref2

Common Purpose ref1
defunding ref1
lockdowns ref1, ref2
masks ref1, ref2, ref3, ref4
police-military state ref1, ref2, ref3
psychopathic personality ref1, ref2, ref3, ref4
reframing ref1
United States ref1, ref2, ref3, ref4
Wokeness ref1

polio ref1

political correctness ref1, ref2, ref3, ref4

political parties ref1, ref2, ref3, ref4

political puppets ref1

pollution ref1, ref2, ref3

post-mortems/autopsies ref1

Postage Stamp Consensus ref1, ref2

pre-emptive programming ref1

Problem-Reaction-Solution ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8

Project for the New American Century ref1, ref2, ref3, ref4

psychopathic personality ref1
Archons ref1
heart energy ref1
lockdowns ref1, ref2, ref3
police ref1, ref2, ref3, ref4
recruitment ref1, ref2
vaccines ref1
wealth ref1
Wetiko ref1, ref2

Psyop (psychological operation), Covid as a ref1, ref2, ref3, ref4, ref5

Pushbackers ref1, ref2, ref3, ref4

pyramid structure ref1, ref2, ref3, ref4

Q

QAnon Psyop ref1, ref2, ref3

R

racism *see also* **Black Lives**
Matter (BLM)
anti-racism industry ref1
class ref1
critical race theory ref1
culture ref1
intersectionality ref1
reverse racism ref1
white privilege ref1, ref2
white supremacy ref1, ref2, ref3, ref4, ref5
Wokeness ref1, ref2, ref3
radiation ref1, ref2
randomness, illusion of ref1, ref2, ref3
reality ref1, ref2, ref3
reframing ref1, ref2
change agents ref1, ref2
children ref1
climate change ref1
Common Purpose leadership programme ref1, ref2
contradictory rules ref1
enforcers ref1
masks ref1, ref2
NLP and the Delphi technique ref1
police ref1
Wetiko factor ref1
Wokeness ref1, ref2
religion *see also* particular religions
alien invasions ref1

Archons ref1, ref2
consciousness ref1, ref2
control, system of ref1, ref2, ref3
criticism, prohibition on ref1
five senses ref1
good and evil, war between ref1
hidden non-human forces ref1, ref2
Sabbatians ref1
save me syndrome ref1
Wetiko ref1
Wokeness ref1
repetition and mind control ref1, ref2, ref3
reporting/snitching, encouragement of ref1, ref2
Reptilians/Grey entities ref1
rewiring the mind ref1
Rivers, Thomas Milton ref1, ref2
Rockefeller family ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9
Rockefeller Foundation documents ref1, ref2, ref3, ref4
Roman Empire ref1
Rothschild family ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9
RT-PCR tests ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8
Russia
collusion inquiry in US ref1
Russian Revolution ref1, ref2
Sabbatians ref1

S

Sabbantian-Frankism ref1, ref2
anti-Semitism ref1, ref2
banking and finance ref1, ref2, ref3
China ref1, ref2
Israel ref1, ref2, ref3, ref4, ref5

Judaism ref1, ref2, ref3, ref4, ref5
Lucifer ref1
media ref1, ref2
Nazis ref1, ref2
QAnon ref1
Rothschilds ref1, ref2, ref3, ref4, ref5, ref6
Russia ref1
Saudi Arabia ref1
Silicon Valley ref1
Sumer ref1
United States ref1, ref2, ref3
Wetiko factor ref1
Wokeness ref1, ref2, ref3

SAGE (Scientific Advisory Group for Emergencies) ref1, ref2, ref3, ref4

SARS-1 ref1

SARs-CoV-2 ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8

Satan/Satanism ref1, ref2, ref3, ref4, ref5, ref6, ref7

satellites in low-orbit ref1

Saudi Arabia ref1

Save Me Syndrome ref1

scapegoating ref1

Schwab, Klaus ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10, ref11, ref12

science, manipulation of ref1

self-identity ref1, ref2, ref3, ref4

self-respect, attacks on ref1

September 11, 2001, terrorist attacks on United States ref1, ref2, ref3, ref4

77th Brigade of UK military ref1, ref2, ref3

Silicon Valley/tech giants ref1, ref2, ref3, ref4, ref5, ref6 *see also* **Facebook**

Israel ref1
Sabbatians ref1
technocracy ref1
Wetiko factor ref1
Wokeness ref1
simulation hypothesis ref1, ref2, ref3, ref4, ref5
Smart Grid ref1, ref2, ref3
artificial intelligence (AI) ref1
China ref1, ref2
control centres ref1
the Field ref1
Great Reset ref1
Human 2.0 ref1, ref2
Israel ref1, ref2
vaccines ref1
Wetiko factor ref1
social disapproval ref1
social distancing *and* **isolation** ref1, ref2, ref3
abusive relationships ref1, ref2
children ref1
flats and apartments ref1
heart issues ref1
hugs ref1
Internet ref1
masks ref1
media ref1
older people ref1, ref2
one-metre (three feet) rule ref1
rewiring the mind ref1
simulation, universe as a ref1
SPI-B ref1
substance abuse ref1

suicide and self-harm ref1, ref2, ref3, ref4, ref5
technology ref1
torture, as ref1, ref2
two-metre (six feet) rule ref1
women ref1
social justice ref1, ref2, ref3, ref4
social media *see also* **Facebook bans on alternative views** ref1
censorship ref1, ref2, ref3, ref4, ref5, ref6
children ref1
emotion ref1
perception ref1
private messages ref1
Twitter ref1, ref2, ref3, ref4, ref5, ref6, ref7
Wetiko factor ref1
YouTube ref1, ref2, ref3, ref4, ref5
Soros, George ref1, ref2, ref3, ref4, ref5, ref6
Spain ref1
SPI-B (Scientific Pandemic Insights Group on Behaviours) ref1, ref2, ref3, ref4
spider and the web ref1, ref2, ref3, ref4
Starmer, Keir ref1
Statute Law ref1
Steiner, Rudolf ref1, ref2, ref3
Stockholm syndrome ref1
streptomycin ref1
suicide and self-harm ref1, ref2, ref3, ref4, ref5
Sumer ref1, ref2
Sunstein, Cass ref1, ref2, ref3
swine flu (H1N1) ref1, ref2, ref3
synchronicity ref1
synthetic biology ref1, ref2, ref3, ref4
synthetic meat ref1, ref2

T

technology *see also* **artificial intelligence (AI); Internet;**
social media addiction ref1, ref2, ref3, ref4
Archons ref1, ref2
the cloud ref1, ref2, ref3, ref4, ref5, ref6, ref7
cyber-operations ref1
cyberwarfare ref1
radiation ref1, ref2
social distancing and isolation ref1
technocracy ref1
Tedros Adhanom Ghebreyesus ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10, ref11, ref12, ref13
telepathy ref1
Tenpenny, Sherri ref1
Tesla, Nikola ref1
testosterone levels, decrease in ref1
testing for Covid-19 ref1, ref2
anal swab tests ref1
cancer ref1
China ref1, ref2, ref3
Corman-Drosten test ref1, ref2, ref3, ref4
death certificates ref1, ref2
fraudulent testing ref1
genetic material, amplification of ref1
Lateral Flow Device (LFD) ref1
PCR tests ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8
vaccines ref1, ref2, ref3
Thunberg, Greta ref1, ref2, ref3
Totalitarian Tiptoe ref1, ref2, ref3, ref4
transgender persons
activism ref1
artificial wombs ref1

censorship ref1
child abuse ref1, ref2
Human 2.0 ref1, ref2, ref3
Wokeness ref1, ref2, ref3, ref4, ref5
women, deletion of rights and status of ref1, ref2
young persons ref1
travel restrictions ref1
Trudeau, Justin ref1, ref2, ref3
Trump, Donald ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10, ref11
Twitter **ref1, ref2, ref3, ref4, ref5, ref6, ref7**

U

UKColumn ref1, ref2
United Nations (UN) ref1, ref2, ref3, ref4, ref5 *see also* **Agenda 21/Agenda 2030 (UN)**
United States ref1, ref2
American Revolution ref1
borders ref1, ref2
Capitol Hill riot ref1, ref2
children ref1
China ref1, ref2
CIA ref1, ref2
Daily Pass tracking system ref1
demographics by immigration, changes in ref1
Democrats ref1, ref2, ref3, ref4, ref5, ref6, ref7
election fraud ref1
far-right domestic terrorists, pushbackers as ref1
Federal Reserve ref1
flu/respiratory diseases statistics ref1
Global Cult ref1, ref2
hand sanitisers, FDA warnings on ref1

immigration, effects of illegal ref1
impeachment ref1
Israel ref1, ref2
Judaism ref1, ref2, ref3
lockdown ref1
masks ref1
mass media ref1, ref2
nursing homes ref1
Pentagon ref1, ref2, ref3, ref4
police ref1, ref2, ref3, ref4
pushbackers ref1
Republicans ref1, ref2
borders ref1, ref2
Democrats ref1, ref2, ref3, ref4, ref5
Russia, inquiry into collusion with ref1
Sabbatians ref1, ref2, ref3
September 11, 2001, terrorist attacks ref1, ref2, ref3, ref4
UFO sightings, release of information on ref1
vaccines ref1
white supremacy ref1, ref2, ref3, ref4
Woke Democrats ref1, ref2

V

vaccines ref1, ref2, ref3
adverse reactions ref1, ref2, ref3, ref4, ref5
Africa ref1
anaphylactic shock ref1, ref2, ref3, ref4
animals ref1, ref2
anti-vax movement ref1, ref2, ref3, ref4, ref5
AstraZeneca/Oxford ref1, ref2, ref3, ref4
autoimmune diseases, rise in ref1, ref2
Big Pharma ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8

bioweapon, as real ref1, ref2
black and ethnic minority communities ref1
blood clots ref1, ref2
Brain Computer Interface (BCI) ref1
care homes, deaths in ref1
censorship ref1, ref2, ref3
chief medical officers and scientific advisers, financial interests of ref1, ref2
children ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10
China ref1, ref2
clinical trials ref1, ref2, ref3, ref4, ref5, ref6
compensation ref1
compulsory vaccinations ref1, ref2, ref3
computer programs ref1
consciousness ref1
cover-ups ref1
creation before Covid ref1
cytokine storm ref1
deaths and illnesses caused by vaccines ref1, ref2, ref3, ref4, ref5
definition ref1
developing countries ref1
digital tattoos ref1
DNA-manipulation ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10
emergency approval ref1, ref2, ref3, ref4, ref5
female infertility ref1
funding ref1
genetic suicide ref1
Global Cult ref1
heart chakras ref1
hesitancy ref1
Human 2.0 ref1, ref2, ref3, ref4
immunity from prosecution ref1, ref2, ref3

implantable technology ref1
Israel ref1
Johnson & Johnson ref1, ref2, ref3, ref4
lockdowns ref1
long-term effects ref1
mainstream media ref1, ref2, ref3, ref4, ref5
masks ref1, ref2, ref3, ref4, ref5
Medicines and Healthcare products Regulatory Agency (MHRA) ref1, ref2
messaging ref1
Moderna ref1, ref2, ref3, ref4, ref5, ref6
mRNA vaccines ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9
nanotechnology ref1, ref2
NHS ref1, ref2, ref3, ref4, ref5
older people ref1, ref2
operating system ref1
passports ref1, ref2, ref3, ref4
Pfizer/BioNTech ref1, ref2, ref3, ref4, ref5, ref6, ref7
polyethylene glycol ref1
pregnant women ref1
psychopathic personality ref1
races, targeting different ref1
reverse transcription ref1
Smart Grid ref1
social distancing ref1
social media ref1
sterility ref1
synthetic material, introduction of ref1
tests ref1, ref2, ref3
travel restrictions ref1
variants ref1, ref2
viruses, existence of ref1
whistle-blowing ref1

WHO ref1, ref2, ref3, ref4
Wokeness ref1
working, vaccine as ref1
young people ref1
Vallance, Patrick ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9
variants ref1, ref2, ref3
vegans ref1
ventilators ref1, ref2
virology ref1, ref2
virtual reality ref1, ref2, ref3
viruses, existence of ref1
visual reality ref1, ref2
vitamin D ref1, ref2
von Braun, Wernher ref1, ref2

W

war-zone hospital myths ref1
waveforms ref1, ref2
wealth ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9 ref10, ref11
wet market conspiracy ref1
Wetiko factor ref1
alcoholism and drug addiction ref1
anti-human, why Global Cult is ref1
Archons ref1, ref2, ref3, ref4
artificial intelligence (AI) ref1
Big Pharma ref1, ref2
children ref1
China ref1
consciousness ref1, ref2
education ref1
Facebook ref1

fear ref1, ref2
frequency ref1, ref2
Gates ref1, ref2
Global Cult ref1, ref2
heart ref1, ref2
lockdowns ref1
masks ref1
Native American concept ref1
psychopathic personality ref1, ref2
reframing/retraining programmes ref1
religion ref1
Silicon Valley ref1
Smart Grid ref1
smartphone addiction ref1, ref2
social media ref1
war ref1, ref2
WHO ref1
Wokeness ref1, ref2, ref3
Yaldabaoth ref1, ref2, ref3, ref4

whistle-blowing ref1, ref2, ref3, ref4, ref5, ref6, ref7

white privilege ref1, ref2

white supremacy ref1, ref2, ref3, ref4, ref5

Whitty, Christopher ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10

'who benefits' ref1

Wi-Fi ref1, ref2, ref3, ref4

Wikipedia ref1, ref2

Wojcicki, Susan ref1, ref2, ref3, ref4, ref5, ref6, ref7

Wokeness
Antifa ref1, ref2, ref3, ref4
anti-Semitism ref1
billionaire social justice warriors ref1, ref2, ref3

Capitol Hill riot ref1, ref2
censorship ref1
Christianity ref1
climate change hoax ref1, ref2
culture ref1
education, control of ref1
emotion ref1
facts ref1
fascism ref1, ref2, ref3
Global Cult ref1, ref2, ref3, ref4
group-think ref1
immigration ref1
indigenous people, solidarity with ref1
inversion ref1, ref2, ref3
left, hijacking the ref1, ref2
Marxism ref1, ref2, ref3
mind control ref1
New Woke ref1
Old Woke ref1
Oneness ref1
perceptual programming ref1
Phantom Self ref1
police ref1
defunding the ref1
reframing ref1
public institutions ref1
Pushbackers ref1, ref2, ref3
racism ref1, ref2, ref3
reframing ref1, ref2
religion, as ref1
Sabbatians ref1, ref2, ref3
Silicon Valley ref1
social justice ref1, ref2, ref3, ref4

transgender ref1, ref2, ref3, ref4, ref5
United States ref1, ref2
vaccines ref1
Wetiko factor ref1, ref2, ref3
young people ref1, ref2, ref3
women, deletion of rights and status of ref1, ref2
World Economic Forum (WEF) ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9
World Health Organization (WHO) ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9
AIDs/HIV ref1
amplification cycles ref1
Big Pharma ref1, ref2, ref3
cooperation in health emergencies ref1
creation ref1, ref2
fatality rate ref1
funding ref1, ref2, ref3
Gates ref1
Internet ref1
lockdown ref1
vaccines ref1, ref2, ref3, ref4
Wetiko factor ref1
world number 1 (masses) ref1, ref2
world number 2 ref1
Wuhan ref1, ref2, ref3, ref4, ref5, ref6, ref7 ref8

Y

Yaldabaoth ref1, ref2, ref3, ref4, ref5, ref6
Yeadon, Michael ref1, ref2, ref3, ref4
young people *see also* **children** addiction to technology ref1
Human 2.0 ref1
vaccines ref1, ref2

Wokeness ref1, ref2, ref3
YouTube ref1, ref2, ref3, ref4, ref5
WHO 548

Z

Zaks, Tal ref1
Zionism ref1, ref2, ref3
Zuckerberg, Mark ref1, ref2, ref3, ref4, ref5, ref6, ref7, ref8, ref9, ref10, ref11, ref12
Zulus ref1

ICKONIC
THE ALTERNATIVE

Ickonic is something that has been a dream of mine for the last 5 years, growing up around alternative information I have always had a natural interest in what is going on in the World and what could I do to make it better.
Across the range of subjects and positions of influence occupied mainly by people who don't strive to make things better it's the Media that I have always found the most frustrating and fascinating. Mainly because if the Media did their Jobs properly then so much of the negative things happening in the World simply would not be able to happen, because they would be exposed within a heartbeat.
Free Press and the Opportunities that the internet could have given would mean that the Media are able to expose things like never before and hold people to account for their actions. As we all know there are 'Untouchables' that walk among us, people the Media simply won't touch, expose or investigate and that leads to the dark underworlds that infest the establishment the World over.
Well I say enough, it's time for something different, a different kind of Media, where no one is off limits from exposing and investigating. All we're interested in at Ickonic is the truth of what is really going on in the World on whichever subject we're covering.
We hope you enjoy what we have created and take something away from the platform, we aim to deliver information that's informative and most importantly self-empowering, you're not a little person, you're part of something much bigger than that and its time we as a collective race began to understand that and look to the future as ours to take.

It's time...

Jaymie Icke - Founder Ickonic Alternative Media.

SIGN UP NOW AT ICKONIC.COM

DAVID ICKE

THE ANSWER

We live in extraordinary times with billions bewildered and seeking answers for what is happening. David Icke, the man who has been proved right again and again, has spent 30 years uncovering the truth behind world affairs and in a stream of previous books he predicted current events.

The Answer will change your every perception of life and the world and set you free of the illusions that control human society. There is nothing more vital for our collective freedom than humanity becoming aware of what is in this book.

Available now at davidicke.com.

THE TRIGGER

THE LIE THAT CHANGED THE WORLD – WHO REALLY DID IT AND WHY

DAVID ICKE

EVERYTHING YOU NEED TO KNOW

BUT HAVE NEVER BEEN TOLD

DAVID ICKE

DAVIDICKE.COM

DAVID ICKE STORE

LATEST NEWS ARTICLES

DAVID ICKE VIDEOS

WEEKLY DOT-CONNECTOR PODCASTS

LIVE EVENTS

WWW.DAVIDICKE.COM

THE LIFE STORY OF DAVID ICKE

RENEGADE

THE FEATURE LENGTH FILM

/ˈren·ɪˌɡeɪd/

noun

A person who behaves in a rebelliously unconventional manner.

AVAILABLE NOW AT DAVIDICKE.COM

2 NEW BOOKS
BY NEIL HAGUE
ORION'S DOOR
SYMBOLS OF CONSCIOUSNESS & BLUEPRINTS OF CONTROL
- THE STORY OF ORION'S INFLUENCE OVER HUMANITY
CUTTING EDGE VISIONARY ART
& UNIQUE ILLUSTRATED BOOKS
THE DR. COVID UNIVERSE
ADVENTURES IN CLOWNLAND
NEIL HAGUE
FOR
BOOKS, PRINTS & T-SHIRTS
VISIT:
NEILHAGUEBOOKS.COM
OR NEILHAGUE.COM

Before you go ...

For more detail, background and evidence about the subjects in *Perceptions of a Renegade Mind* – and so much more – see my others books including *And The Truth Shall Set You Free; The Biggest Secret; Children of the Matrix; The David Icke Guide to the Global Conspiracy; Tales from the Time Loop; The Perception Deception; Remember Who You Are; Human Race Get Off Your Knees; Phantom Self; Everything You Need To Know But Have Never Been Told, The Trigger and The Answer*.

You can subscribe to the fantastic new Ickonic media platform where there are many hundreds of hours of cutting-edge information in videos, documentaries and series across a whole range of subjects which are added to every week. This includes my 90 minute breakdown of the week's news every Friday to explain *why* events are happening and to what end.