Packt>

BEGINNING MODERN C# AND .NET DEVELOPMENT

Twitter: @OnurGumusDev

LinkedIn :https://www.linkedin.com/in/onurgumus

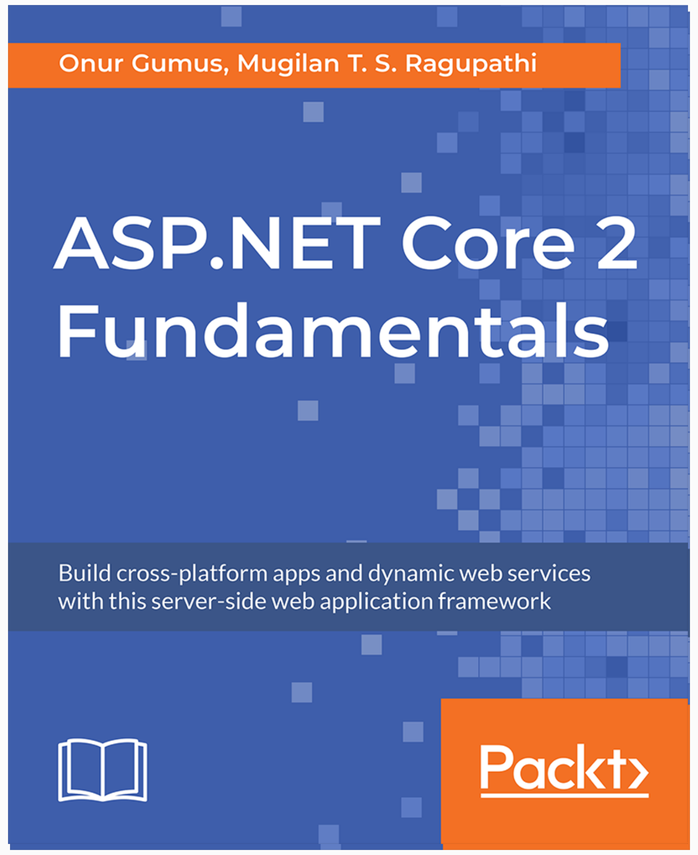
Source Code:
https://github.com/OnurGumus/BeginningCSharp

- A lead software engineer based in Dubai.
- Functional programming and .NET enthusiast

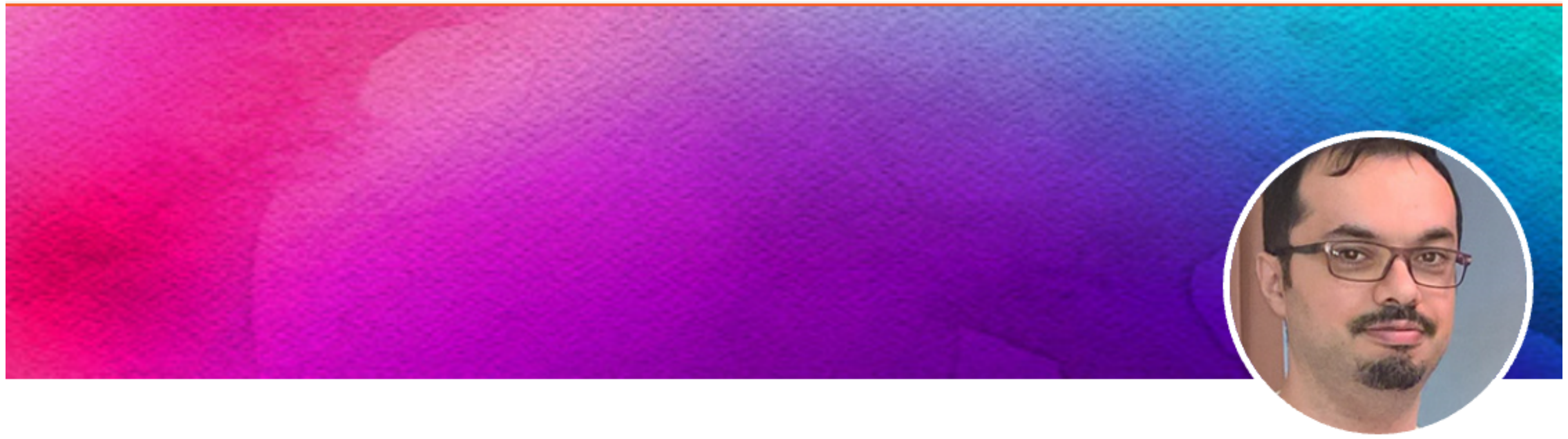
Onur Gumus
Learning Functional Programming with F#
Get started on building end-to-end web applications with F#
Packt>

Onur Gumus
Functional Application Designing
Start building end-to-end applications with F#
Packt>

Onur Gumus
End-to-End Real-World Application Development with F#
Build efficient web applications with F#
Packt>



Onur Gumus, Mugilan T. S. Ragupathi
ASP.NET Core 2 Fundamentals
Build cross-platform apps and dynamic web services with this server-side web application framework
Packt>



March 27 & 28, 2019

Mastering C# 8.0 and .NET Core 3.0

Presented by **Onur Gumus**

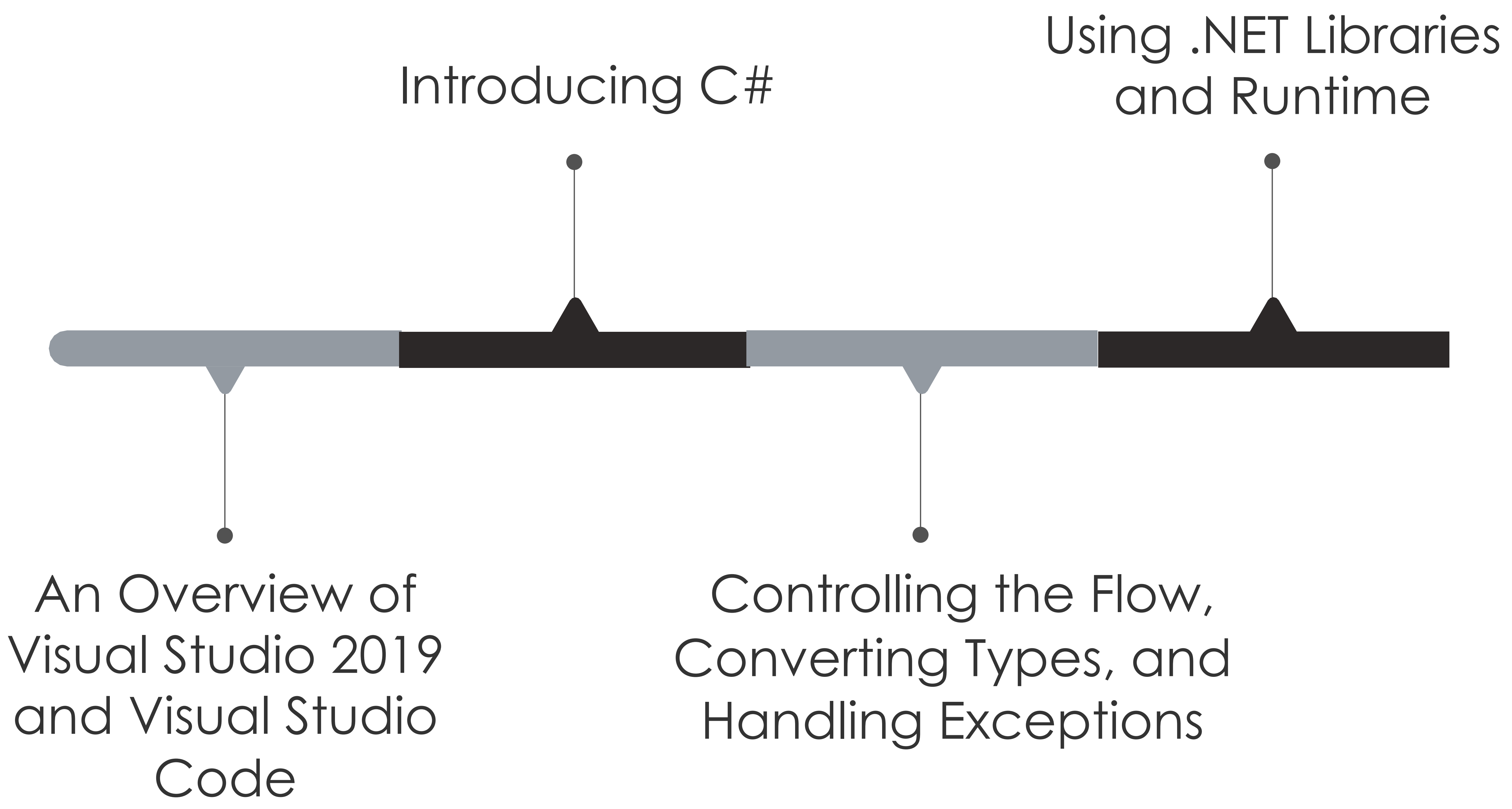
This course will not...

- make you an expert C# developer immediately.
- cover all the features of C#.
- cover optimizing, testing, deployment in detail.

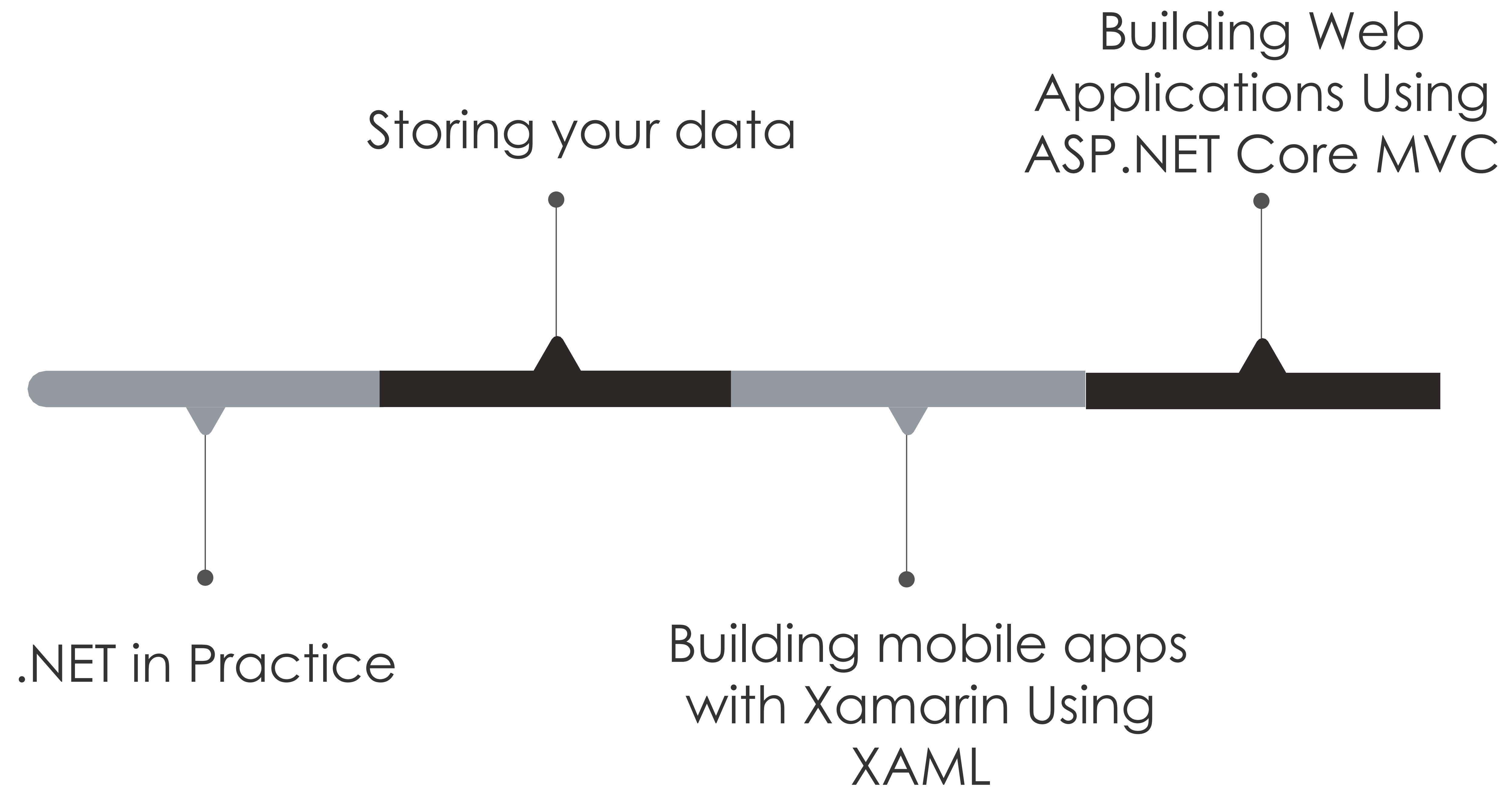
This course will …

- give you the initial push for C# and .NET so that you can continue on your own.
- make you aware of the concepts that you will know what to look for.
- provide you some hands on practice on C# and .NET Core.

Roadmap Day 1


Using .NET Libraries and Runtime
Introducing C#
An Overview of Visual Studio 2019 and Visual Studio Code
Controlling the Flow, Converting Types, and Handling Exceptions

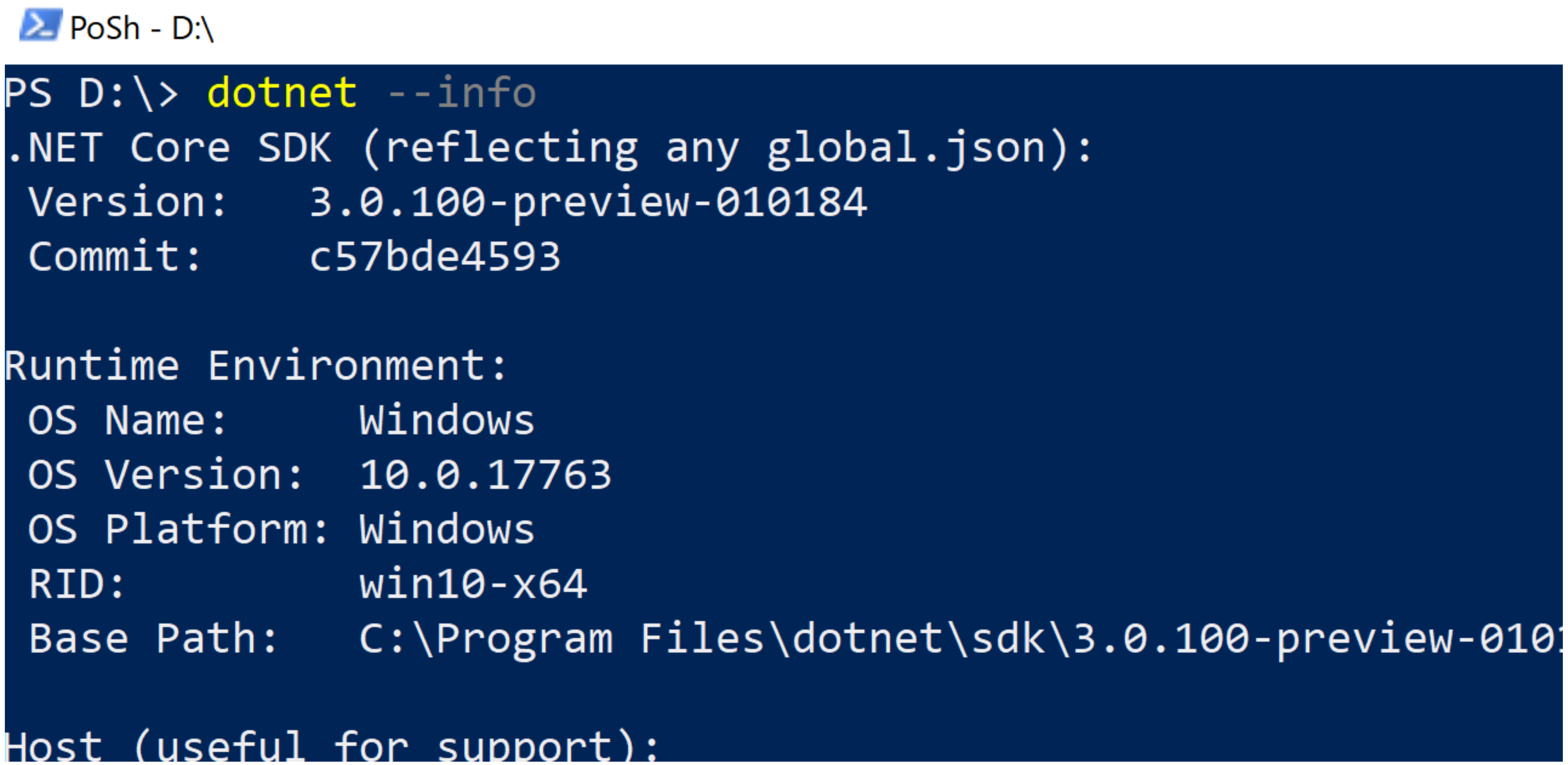
Roadmap Day 2


Storing your data
Building Web Applications Using ASP.NET Core MVC
.NET in Practice
Building mobile apps with Xamarin Using XAML

Prerequisites

- .NET Core 3.0 SDK https://dotnet.microsoft.com/download/dotnet-core/3.0
- For windows users: Visual Studio 2019 , (Visual Studio 2017 is OK) (optional)
  For mobile, consider: Xamarin and UWP workload
  For Web, consider : ASP.NET Workload
- For mac users: Visual Studio for Mac (optional)
- For everyone: Visual Studio Code (optional) and C# for Visual Studio Code extension

.NET Core SDK


PoSh - D:\

```
PS D:\> dotnet --info
.NET Core SDK (reflecting any global.json):
 Version:   3.0.100-preview-010184
 Commit:    c57bde4593

Runtime Environment:
 OS Name:     Windows
 OS Version:  10.0.17763
 OS Platform: Windows
 RID:         win10-x64
 Base Path:   C:\Program Files\dotnet\sdk\3.0.100-preview-010

Host (useful for support):
```

Lesson 1:

An Overview of Visual Studio 2019 and Visual Studio Code

In this lesson you will learn about…

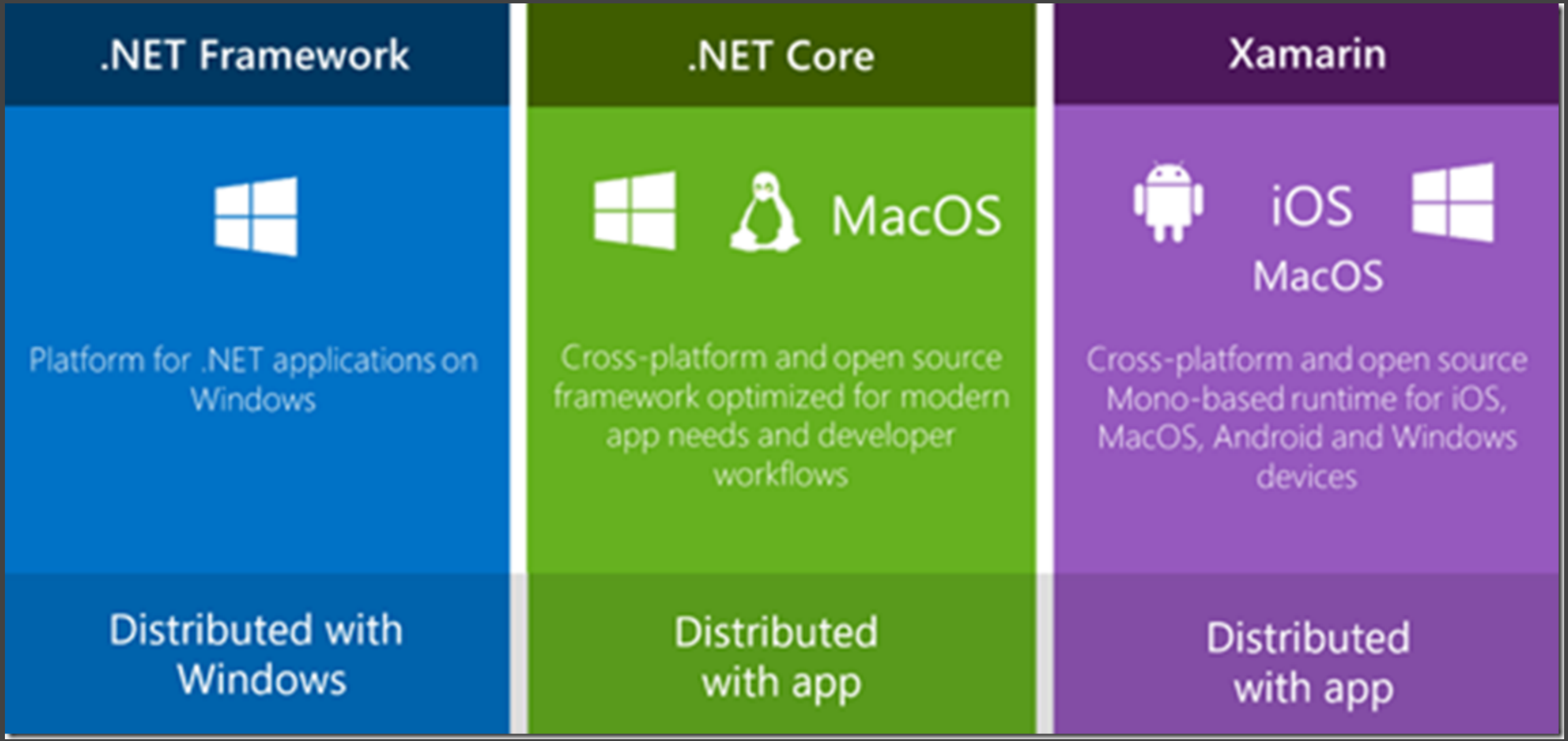
- .NET, .NET Core and .NET Standard.
- fundamentals of Visual Studio 2019
- fundamentals of Visual Studio Code and .NET Core CLI

Section 1.1

.NET, .NET Core and .NET Standard.

An Overview of Visual Studio 2017 and Visual Studio Code

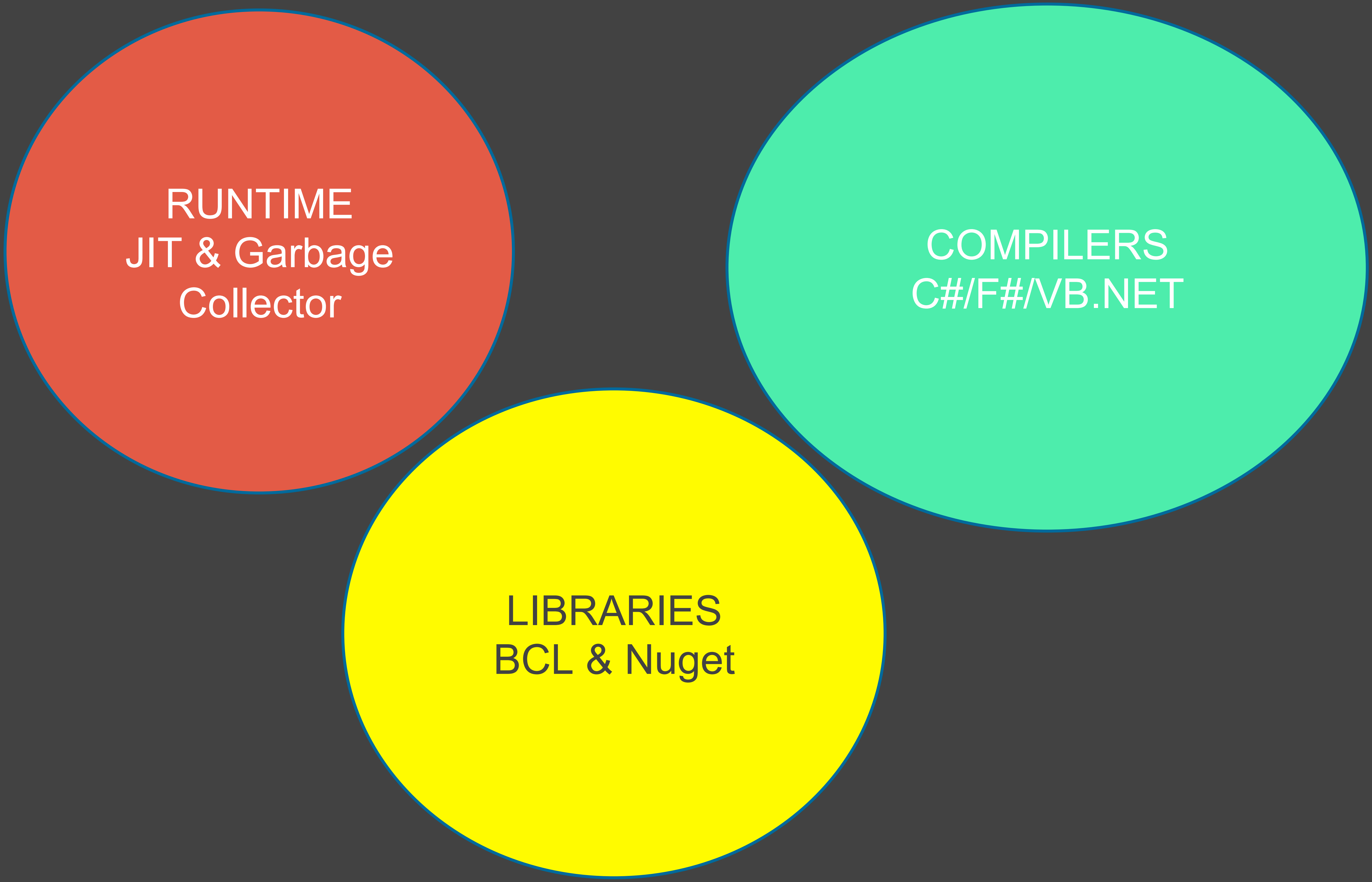
.NET



.NET Framework	.NET Core	Xamarin
Windows	Windows, Linux, MacOS	Android, iOS, Windows, MacOS
Platform for .NET applications on Windows	Cross-platform and open source framework optimized for modern app needs and developer workflows	Cross-platform and open source Mono-based runtime for iOS, MacOS, Android and Windows devices
Distributed with Windows	Distributed with app	Distributed with app

.NET STANDARD (compatible with all)

.NET


RUNTIME
JIT & Garbage
Collector
COMPILERS
C#/F#/VB.NET
LIBRARIES
BCL & Nuget

Future of .NET

… what is also true is that **the rate of innovation in .NET Framework has to slow down** in order to reduce breakage. In that sense, you should generally expect that **most new features will only become available on .NET Core** (and derived platforms, such as Xamarin, Mono, and Unity as they build from the same sources as .NET Core).

Immo Landwerth
Program manager on the .NET team at Microsoft.

Section 1.2

Fundamentals of Visual Studio Code and .NET Core CLI

Visual Studio Code and .Net Core

- Visual Studio Code is a free light weight editor.
- Supports many languages like C#, F#, VB, C++, Python, JavaScript.
- Visual Studio Code is itself written with JavaScript.
- It offers project organization and intellisense.
- .NET Core is a cross platform alternative of .NET.
- It offers wide set of command line tools

dotnet CLI

- `dotnet new sln -o MySolution  //creates a solution and a folder`
- `dotnet new console -o MyConsole //creates a console project and a folder`
- `dotnet new classlib -o MyLib //creates a class lib project and a folder`
- `dotnet new xunit -o MyTest //creates an xunit project and a folder`
- `dotnet sln add <Path_To_Project> //adds given project to the solution`
- `dotnet add <Path_To_Project> reference <Path_To_Target_Project>`
  `//adds given project to as a reference`
- `dotnet add <Path_To_Project> package <Name_of_nuget_packet>`
  `//adds given nuget to the project`

Activity – VSCode and .NET Core CLI Demo

Section 1.3

Fundamentals of Visual Studio 2019

An Overview of Visual Studio 2019 and Visual Studio Code

Visual Studio

- Visual studio is a full featured IDE.
- Supports many languages like C#, F#, VB, C++, Python.
- It has free community edition as well as professional and Enterprise.
- It offers project organization, build tools, intellisense, debugging tools and analyzers.
- Only available to windows.

Activity – VS 2019 Demo

Assessment Question 1

Is .NET Core installation mandatory on target machine during the deployment?

A- YES

B- NO

Slide10

In this lesson you learned…

- Differences between different .NET Runtimes
- How to use Visual Studio Code
- How to use .NET Core CLI and VS Code

Lesson 2: Introducing C#

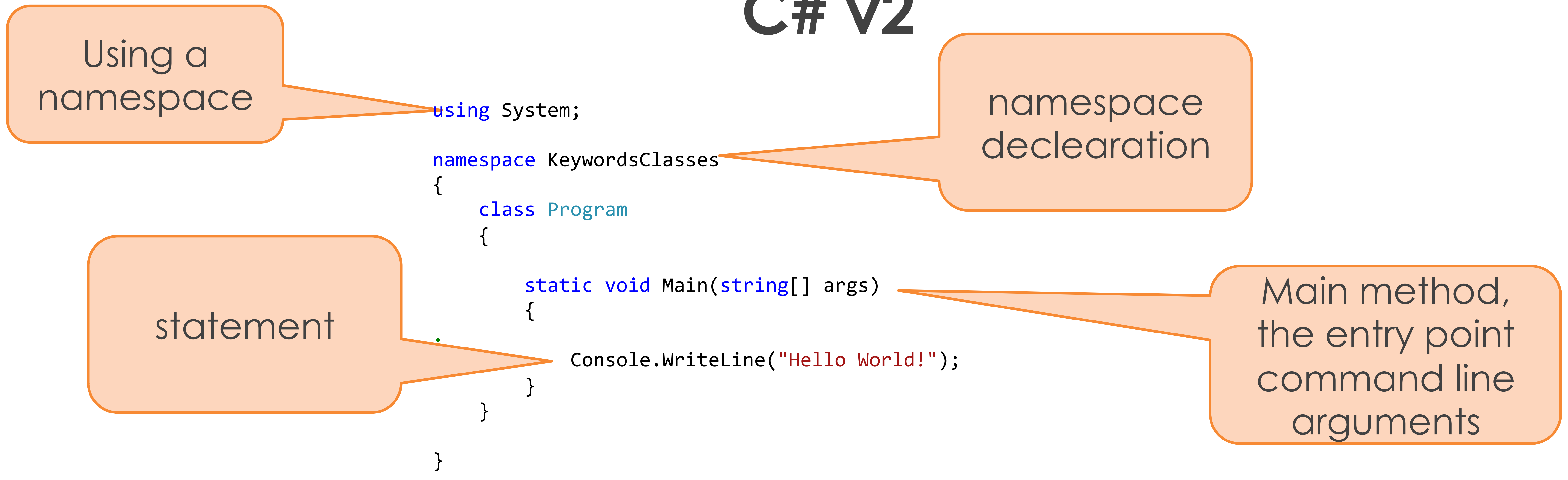
In this lesson you will learn about…

- Exploring the Basics of C#
- Declaring Variables
- Building Console Applications

Section 2.1

Exploring the Basics of C#

C# v2



```
using System;

namespace KeywordsClasses
{
    class Program
    {

        static void Main(string[] args)
        {

            Console.WriteLine("Hello World!");
        }
    }

}
```

C# v2

comment

```
    //This is a comment.
    class Test1
    {

        static int X;

        public string S { get; set; }
    }
}
```

private
Static int

Auto
property

C# v2

```
class Product
{

    public string Name { get; set; }

    public void CalculateTax()
    {

        List<Product> products = new List<Product>();

        using (Stream stream = new MemoryStream())
        {

            if (stream as MemoryStream != null)
            {

            }
        }
    }
    …
    …
```

Auto properties

Generics

using

as

Keywords, Classes Methods demo

Section 2.2

Declaring Variables

Introducing C#

C# v2

string

Also a string via type inference

```
class Program
{
    static void Main(string[] args)
    {
        string s = "Hello";
        int x = 35;
        var s2 = "World";
        var y = 45;
        Test(y);
    }

    public static void Test(int z)
    {

    }
}
```

string is an alias for System.String

int is an alias for System.Int32

Variables Demo

Section 2.3

Building Console Applications

Introducing C#

Activity - Console Application Demo

Assessment Questions

Assessment Question 1

What is type inference and why it is useful?

In this lesson you learned…

- building blocks of C#
- Naming conventions
- Project types

Lesson 3:

Controlling the Flow, Converting Types, and Handling Exceptions

In this lesson you will learn about…

- Basic Control Flow
- Casting and Converting Between Types
- Using Linq
- What are Exceptions?
- Checking for Overflow

Section 3.1

Basic Control Flow

Controlling the Flow, Converting Types, and Handling Exceptions

Control Blocks

new in C# 7

```
if (Console.ReadLine() is var x && x == "hello")
{
    int i = 0;
    do
    {
        Console.WriteLine($"Current number for do while loop is {i}");
        i++;
    } while (i <= 10);

    for (var j = 10; j > 0; j--)
    {
        Console.WriteLine($"Current number for for loop is {j}");
    }
}
```

do - while

string interpolation

foreach

Control Blocks

```
var myArray = new[] { 1, 2, 3, 4, 5, 7, 8, 9, 10 };

foreach (var i in myArray)
{
    switch (i.ToString())
    {
        case "1":
            Console.WriteLine("i is 1");
            break;
        default:

            Console.WriteLine("i is not 1");
            break;
    }
}
```

array initialization
With type inference

foreach -
iterates each
element

Control Blocks

```
public static IEnumerable<char> AlphabetSubset3(char start, char end)
{
    if (start < 'a' || start > 'z')
        throw new ArgumentOutOfRangeException(paramName: nameof(start), message: "start must be
a letter")
    if (end < 'a' || end > 'z')
        throw new ArgumentOutOfRangeException(paramName: nameof(end), message: "end must be a
letter");

    if (end <= start)
        throw new ArgumentException($"{nameof(end)} must be greater than {nameof(start)}");

    return alphabetSubsetImplementation();

    IEnumerable<char>  alphabetSubsetImplementation()
    {
        for (var c = start; c < end; c++)
            yield return c;
    }
}
```

Nested function

yield return as state machine

Control Blocks

```
var values = new object[] { 0, 1, new int[] { 1, 2, 3 }, Array.Empty<int>(), new object[0], "a", null };
foreach (var item in values)
{
    switch (item)
    {
        case 0:
            break;
        case int val:
            sum += val;
            break;
        case IEnumerable<int> subList when subList.Any():
            sum += subList.Sum();
            break;
        case object[] subList2:
            break;
        case null:
            break;
        case object o:
            break;
        //    case var ax:

        default:
            throw new InvalidOperationException("unknown item type");
    }
```

Activity – Control Blocks Demo

Control Blocks

```
var values = new object[] { 0, 1, new int[] { 1, 2, 3 }, Array.Empty<int>(), new object[0], "a", null };
foreach (var item in values)
{
    switch (item)
    {
        case 0:
            break;
        case int val:
            sum += val;
            break;
        case IEnumerable<int> subList when subList.Any():
            sum += subList.Sum();
            break;
        case object[] subList2:
            break;
        case null:
            break;
        case object o:
            break;
        //    case var ax:

        default:
            throw new InvalidOperationException("unknown item type");
    }
```

Section 3.2

Casting and Converting

Controlling the Flow, Converting Types, and Handling Exceptions

Casting and Conversion

```
int i = 5;

double d = i;
Console.WriteLine(d);
d = 5.8;

i = (int)d;

Console.WriteLine(i);
string s = "test";

object o = s;
Console.WriteLine(o);

s = (string)o;
Console.WriteLine(s);
```

Conversion to larger type

Data lost

Casting to base type, no conversion

downcasting

Casting and Conversion

```
class Currency
{
    public int Value { get; }
    public Currency(int value)
    {
        this.Value = value;
    }

    public static implicit operator Currency(int i)
        => new Currency(i);

    public static explicit operator int(Currency c)
        => c.Value;
}
```

Implicit conversion

Explicit conversion

Casting and Conversion

```
int i = 5;
int? j = null;
//implicit casting
j = i;

j = null;
//  throws an exception
//i = (int)j;


Currency c = 5;
int value = (int) c;

 var s = c.Value.ToString();


var v = int.Parse(s);

s = Convert.ToString(v);

v = Convert.ToInt32(s);
if(int.TryParse(s, out var s2))
{
    Console.WriteLine(s2);
}
```

Implicit conversion

Explicit conversion

Implicit conversion

Explicit conversionc

Forced parsing

Try to parse

Casting and Conversion

```
var d = new Derived();

Base b = d;
d = (Derived)b;
I i = d;
b = (Base)i;
d = b as Derived;
if(b is Derived d2)
{
    //use d2
}


int j = 5;
//boxing
object o = j;
//will fail
//j = (int)(long)o;
//unboxing
j = (int)o;
//
```

Notice as keyword

unboxing

Boxing : Value type to object

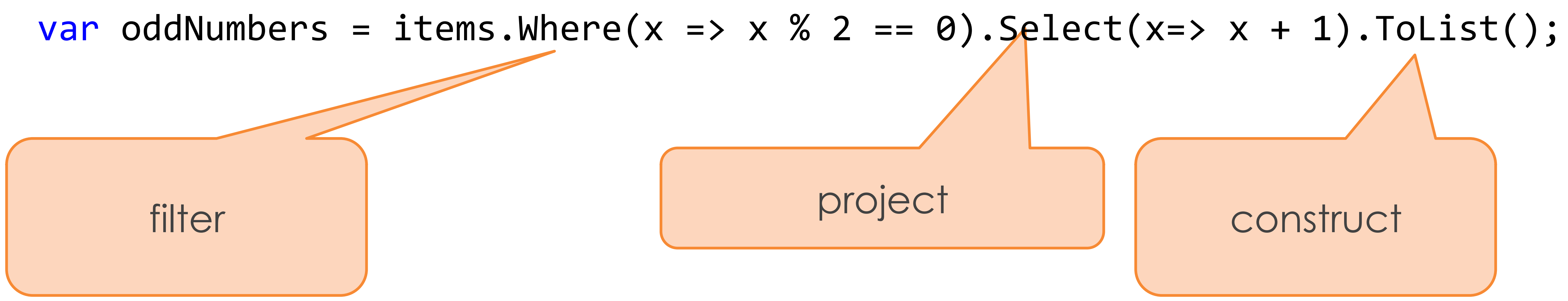
Activity – Casting Demos

Section 3.3

Using Linq

Controlling the Flow, Converting Types, and Handling Exceptions

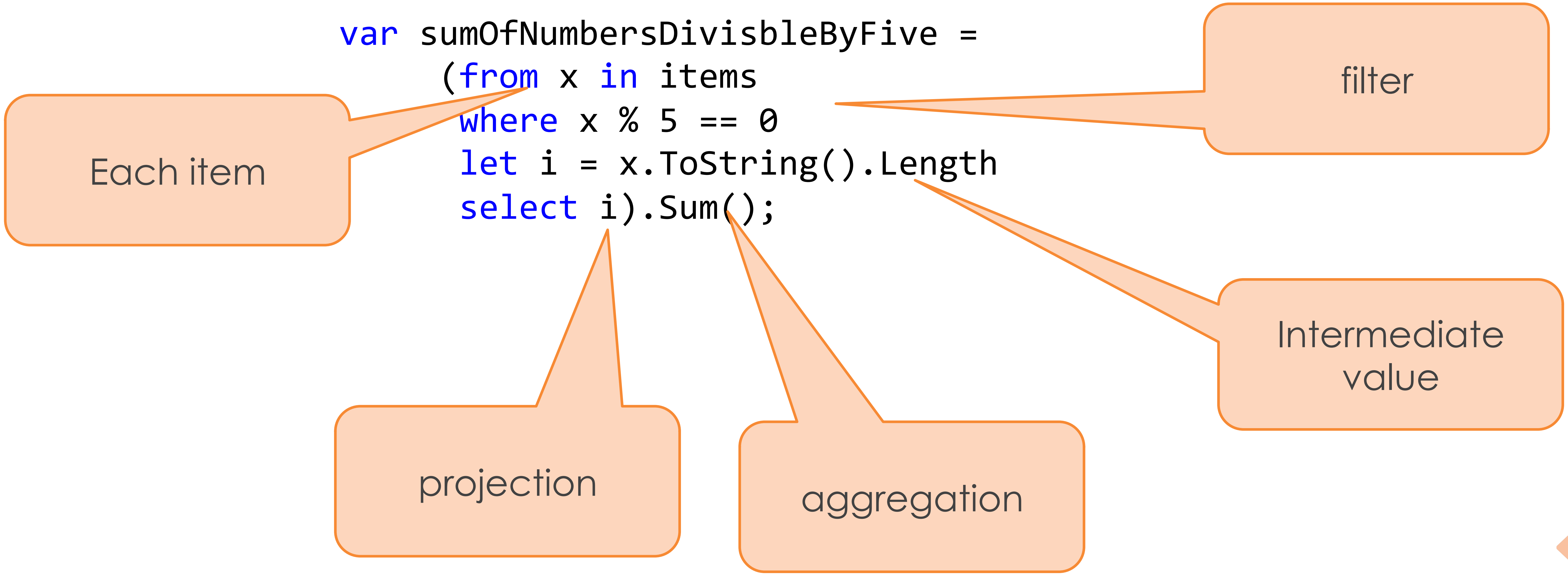
Linq as extension methods



```
var oddNumbers = items.Where(x => x % 2 == 0).Select(x=> x + 1).ToList();
```

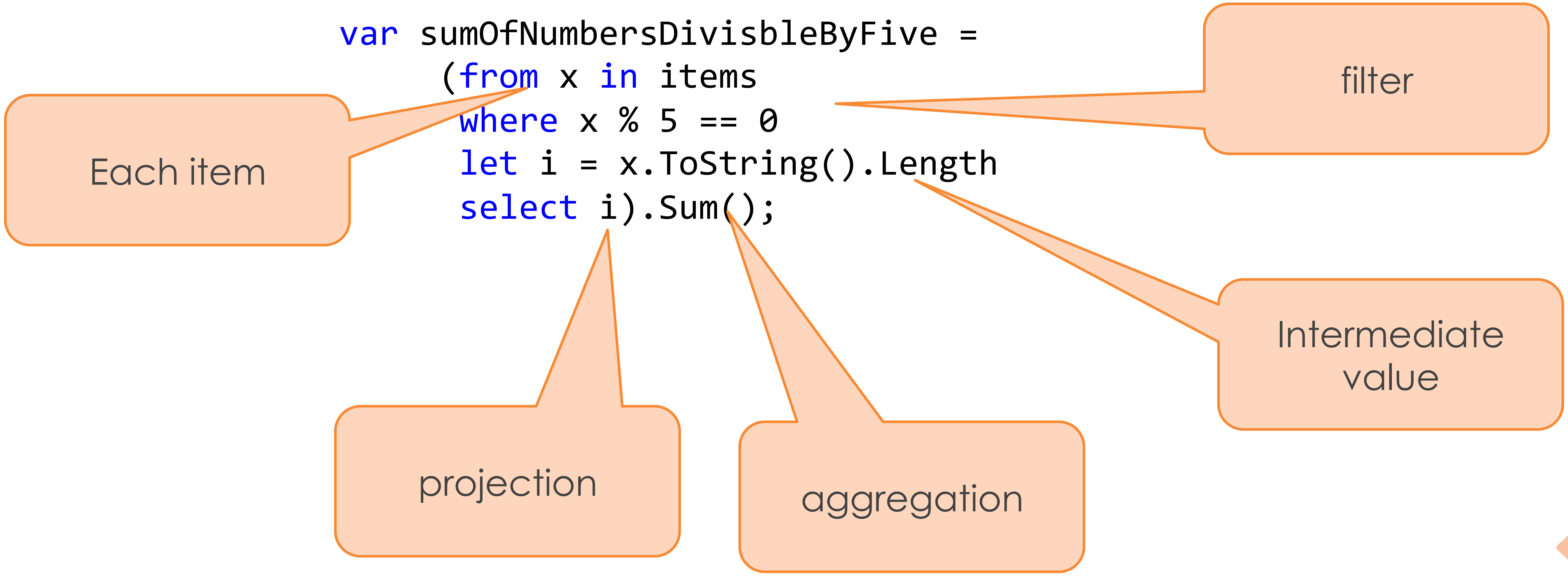
```
Func<int, bool> oddFilter = x => x % 2 == 1;

oddNumbers = items.Where(oddFilter).ToList();
```

Linq as Query



```
var sumOfNumbersDivisbleByFive =
      (from x in items
       where x % 5 == 0
       let i = x.ToString().Length
       select i).Sum();
```

Linq as Query



```
var sumOfNumbersDivisbleByFive =
      (from x in items
       where x % 5 == 0
       let i = x.ToString().Length
       select i).Sum();
```

Linq as Query

Expression Tree

```
var itemsQ = items.AsQueryable();
var multiplicationOfOddnumbers = itemsQ.Where(x => x % 2 == 1)
.Aggregate(1, (x, y) => x * y);
```

Seed

Custom aggregation function

Linq

- Linq stands for language integrated query.
- By using linq you can query memory objects as well as database or web services.
- Linq has two forms either query form or lambda expression form.
- It either takes delegates or expression trees.
- It is possible to write your own linq provider.

Activity – Linq Demo

Section 3.4

What are exceptions?

Controlling the Flow, Converting Types, and Handling Exceptions

Exceptions

- Exceptions leave your code clean for error checks.
- Exceptions don’t occupy your return values.
- Exceptions can carry more information than basic return values.
- Exceptions can’t be ignored.

Exceptions

```
try
{
    var length2 = Length(null);
}
catch (InvalidOperationException e)
{
    Console.WriteLine(e);
}
catch (ArgumentNullException e) when (e.ParamName == "test")
{
    Console.WriteLine(e);
}
catch (ArgumentNullException e) when (e.ParamName == "s")
{
    Console.WriteLine("thrown for s");
    throw;
}
catch (Exception e)
{
    Console.WriteLine(e);
}
finally
{
    Console.WriteLine("finally");
}
```

Conditional exception catching

Catch all exceptions

finally executes no matter what.

Activity – Exceptions Demo 1

Section 3.5

Remaining C# features

Controlling the Flow, Converting Types, and Handling Exceptions

C# v4

```
public IEnumerable<string> GetProductsByname(dynamic s)
{
    var products = new List<Product>();
    return from p in products
           where p.Name == s select (string)s;
}
```

dynamic

C# v5

```
public async Task<IEnumerable<string>> GetProductsBynameAsync()
{
    return await File.ReadAllLinesAsync("Products");
}
```

async and await

C# v6

```
public decimal Price { get; } = 5M;

public  Task<string[]> GetProductsBynameAsync2()
    => ReadAllLinesAsync("Products");

public void ChangeName(Product p)
    => this.Name = p?.Name;


public string Description => $"{this.Name}";
}
```

Direct property initialization

Expression bodied members

null conditional operator

string interpolation

C# v6

```
public decimal CalculateTax2()
{
    try
    {
        return this.Price * 0.1M;
    }

    catch (Exception e) when (e.Message.Contains(nameof(Product)))
    {

    }
    return this.Price;
}
```

Exception filter

nameof

C# v7

```
static async Task Main(string[] args)
{ …
```

Async main

C# v7

```
var result = list.Select(c => (c.Length, c.First())).First();
Console.WriteLine(result.Length);
Console.WriteLine(result.Item2);
var x = 1_000_000;
```

Infer tuple names

Digit seperators

C# v7

```
static readonly Person[] People = new Person[] { new Person() { Name = "Naruto" } };

public static ref Person GetContactInformation(string fname, string lname)
{
     return ref People[0];
}


public static void ConsumeRef()
{
    ref var p = ref GetContactInformation("foo", "bar");
    p = new Person { Name = "Jiraya" };
}
```

ref return In C# ref works for :
- variables (local or parameters)
- fields
- array locations

Changes the value inside people array

C# v7

```
static async Task<List<string>> Main(CancellationToken token = default)
{
    await Task.Delay(100);
    return new List<string> { "a", "b", "c" };
}
```

Default expressions

```
if (int.TryParse(input, out var answer))
    Console.WriteLine(answer);
```

out variables

```
if (item is int val) sum += val;
```

is expressions

C# v7

```
public ValueTask<int> CachedFunc()
{
    return (cache) ? new ValueTask<int>(cacheResult) : new ValueTask<int>(LoadCache());
}
private bool cache = false;
private int cacheResult;
private async Task<int> LoadCache()
{
    // simulate async work:
    await Task.Delay(100);
    cacheResult = 100;
    cache = true;
    return cacheResult;
}
```

Avoid allocation if cached

C# v7

```
int readonlyArgument = 44;
InArgExample(readonlyArgument);
Console.WriteLine(readonlyArgument);     // value is still 44

void InArgExample(in int number)
{
    // Uncomment the following line to see error CS8331
    //number = 19;
}
```

in same as readonly ref

C# v8

```
static void Main(string[] args)
{
    using var options = Parse(args);
    if (options["verbose"]) { WriteLine("Logging..."); }

}
```

using keyword without curly braces

C# v8

```
class Point
{
    public int X { get; }
    public int Y { get; }
    public Point(int x, int y) => (X, Y) = (x, y);
    public void Deconstruct(out int x, out int y) => (x, y) = (X, Y);
}

static string Display(object o)
{
    switch (o)
    {
        case Point p when p.X == 0 && p.Y == 0:
            return "origin";
        case Point p:
            return $"({p.X}, {p.Y})";
        default:
            return "unknown";
    }
}


static string Display2(object o) => o switch
{
            Point { X: 0, Y: 0 } p => "origin",
            Point { X: var x, Y: var y } p => $"({x}, {y})",
            _ => "unknown"
};
```

Deconstructor

C# v8

```
foreach (var name in names[1..4])

foreach (var name in names[1..^1])

await foreach (var name in GetNamesAsync())
```

Ranges

Asynchronous streams

C# v8

```
string text1 = null;
// Warning: Cannot convert null to non-nullable reference
string? text2 = null;
string text3 = text2;
// Warning: Possible null reference assignment
Console.WriteLine(text2.Length );
// Warning: Possible dereference of a null reference
if(text2 != null) { Console.WriteLine(text2.Length); }
// Allowed given check for null
```

Lab1 – Guess My Number

Guess My Number Requirements

Write a game as below.

Computer picks up a random number between 1 and 100.

It asks you to guess the number from Console.

You enter the number to console and hit enter.

If your guess is smaller than the actual number the computer prints "Sorry too small!"
If your guess is larger than the actual number the computer prints "Sorry too big!"

As long as your number is incorrect it keeps asking for new guesses.
If your guess is equal to the actual number the computer prints :
"Congrats! You have found in N attempts!"

Asessment Questions

Assessment Question 1

Why use exceptions instead of return codes?

Assessment Question 2

Which of the following works without losing data?

A-) int x = (int)5.4
B-) int x = 5.4
C-) float x = 0.5f
D-) double y = 0.1

In this lesson you learned…

- how to handle flow
- exceptions
- using LINQ to query your data
- type conversions and casting

Lesson 4:

Using .NET Libraries and Runtime

In this lesson you will learn about…

- Using assemblies and namespaces
- Debugging
- Storing Data with Collections
- Asynchronous programming and tasks.
- Monitoring Performance and Resource Usage

Section 4.1

Using Assemblies and Namespaces

Using .NET Libraries and Runtime

Assemblies, NuGet Packages, and Platforms

- Assemblies are output of your projects.
- They are dll files.
- Assemblies cannot have cyclic dependencies.
- Namespaces only goal is to prevent naming clashes.
- Nuget packages can wrap on or more assemblies and allow you to distribute and re use it for other projects along with a sophisticated dependency algorithm

Packt›

Activity – Assemblies and namespaces demo

Section 4.2

Debugging tools

Using .NET Libraries and Runtime

Activity – Debugging demo

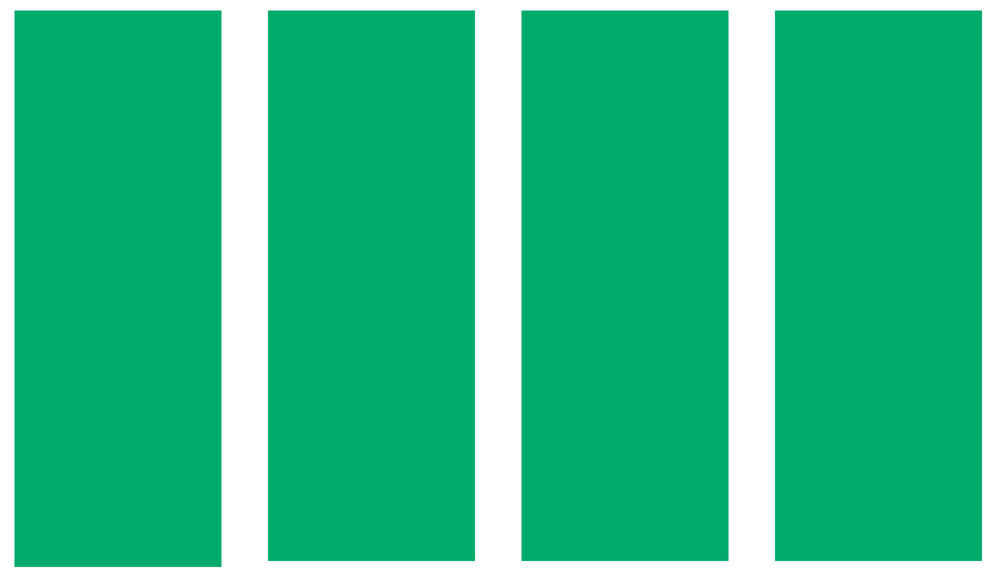
Packt›

Section 4.3

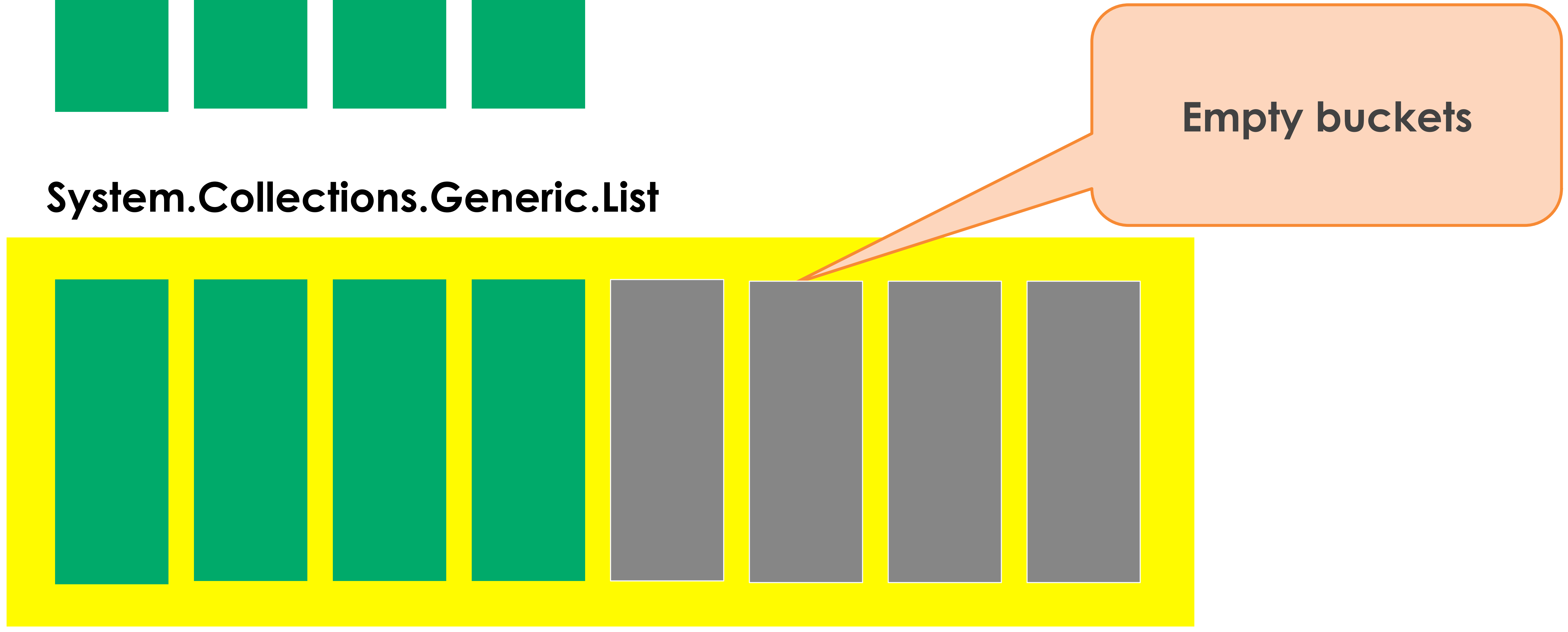
Storing data with collections

Using .NET Libraries and Runtime

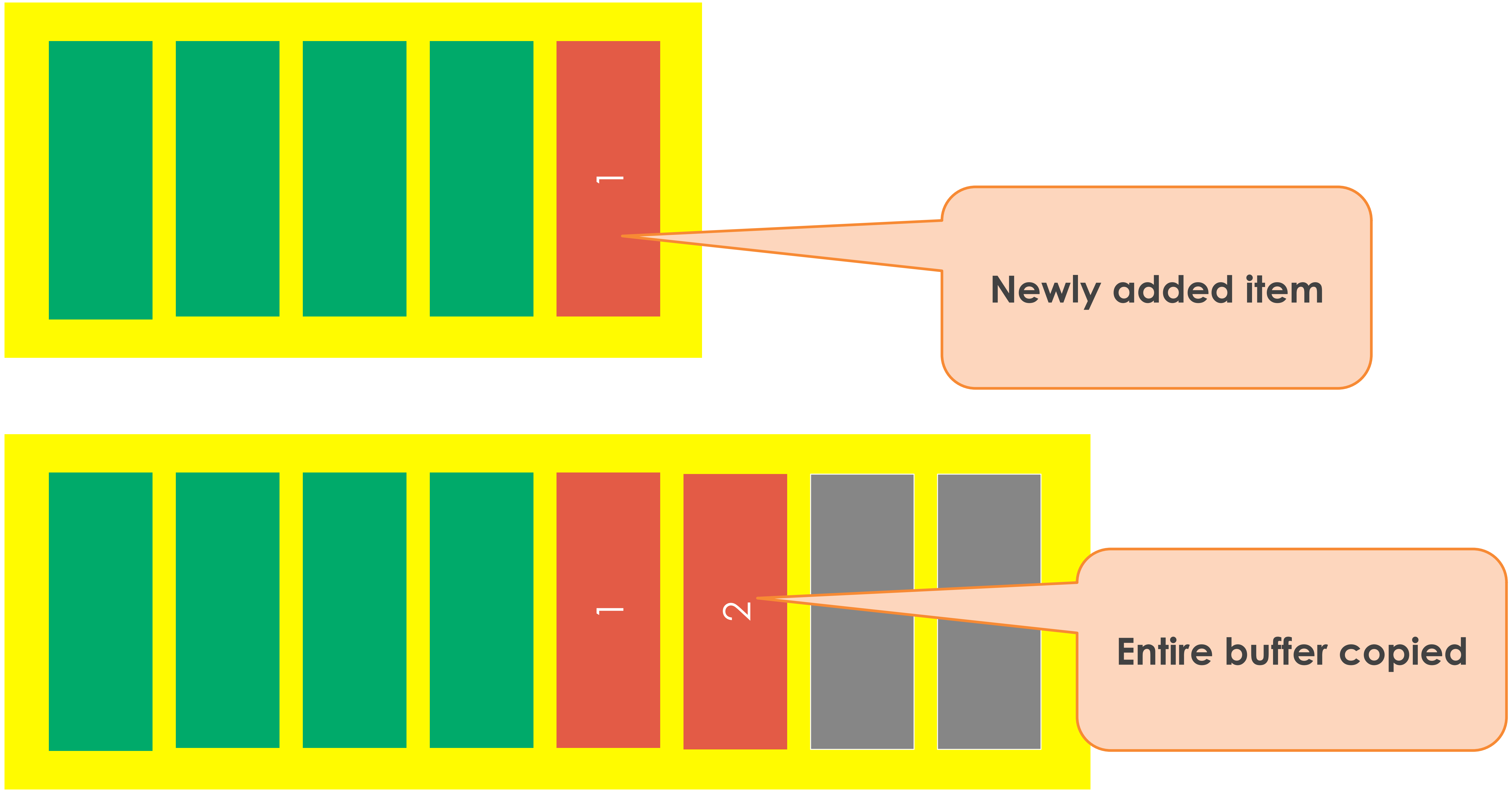
Array



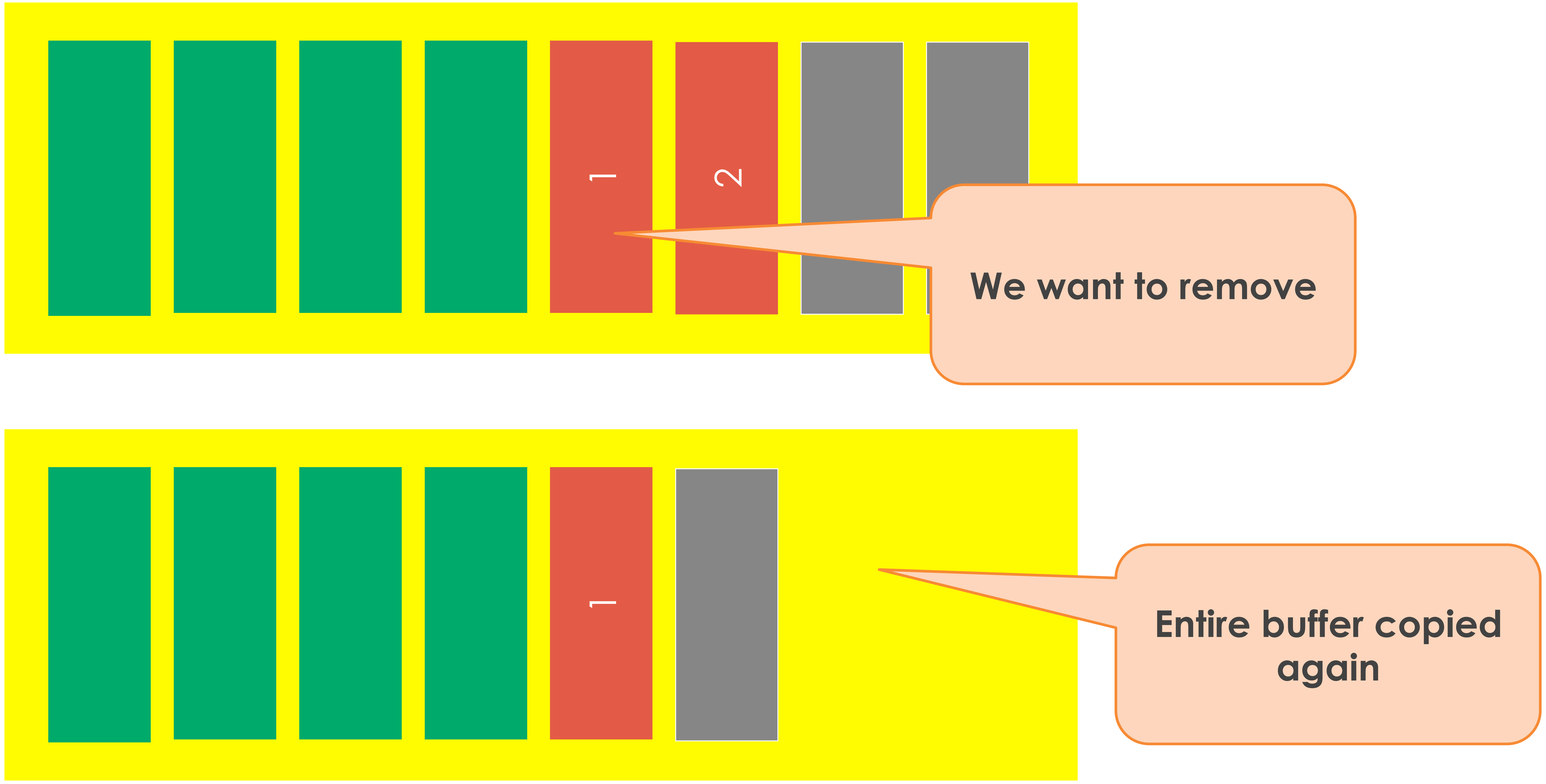
System.Collections.Generic.List



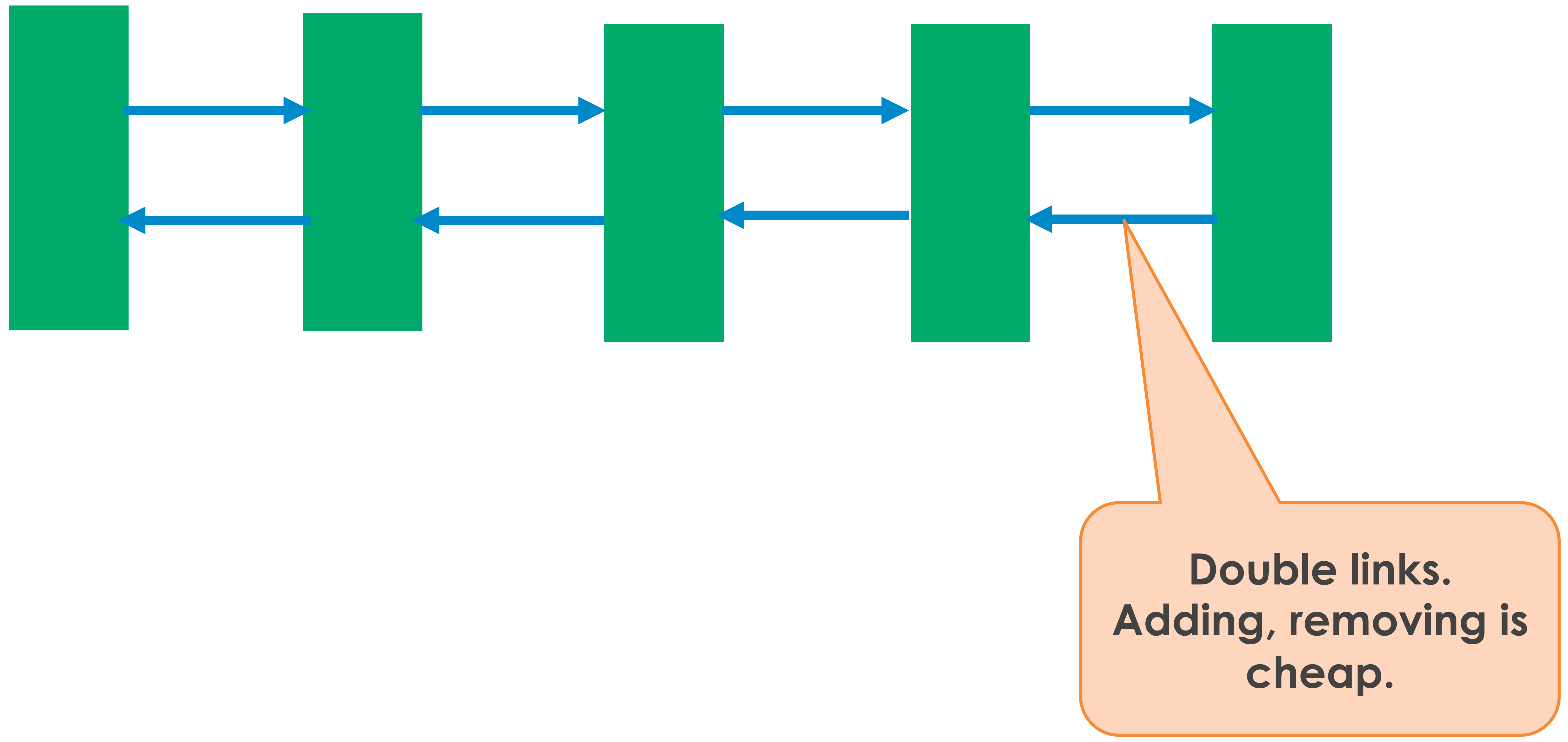
System.Collections.Generic.List



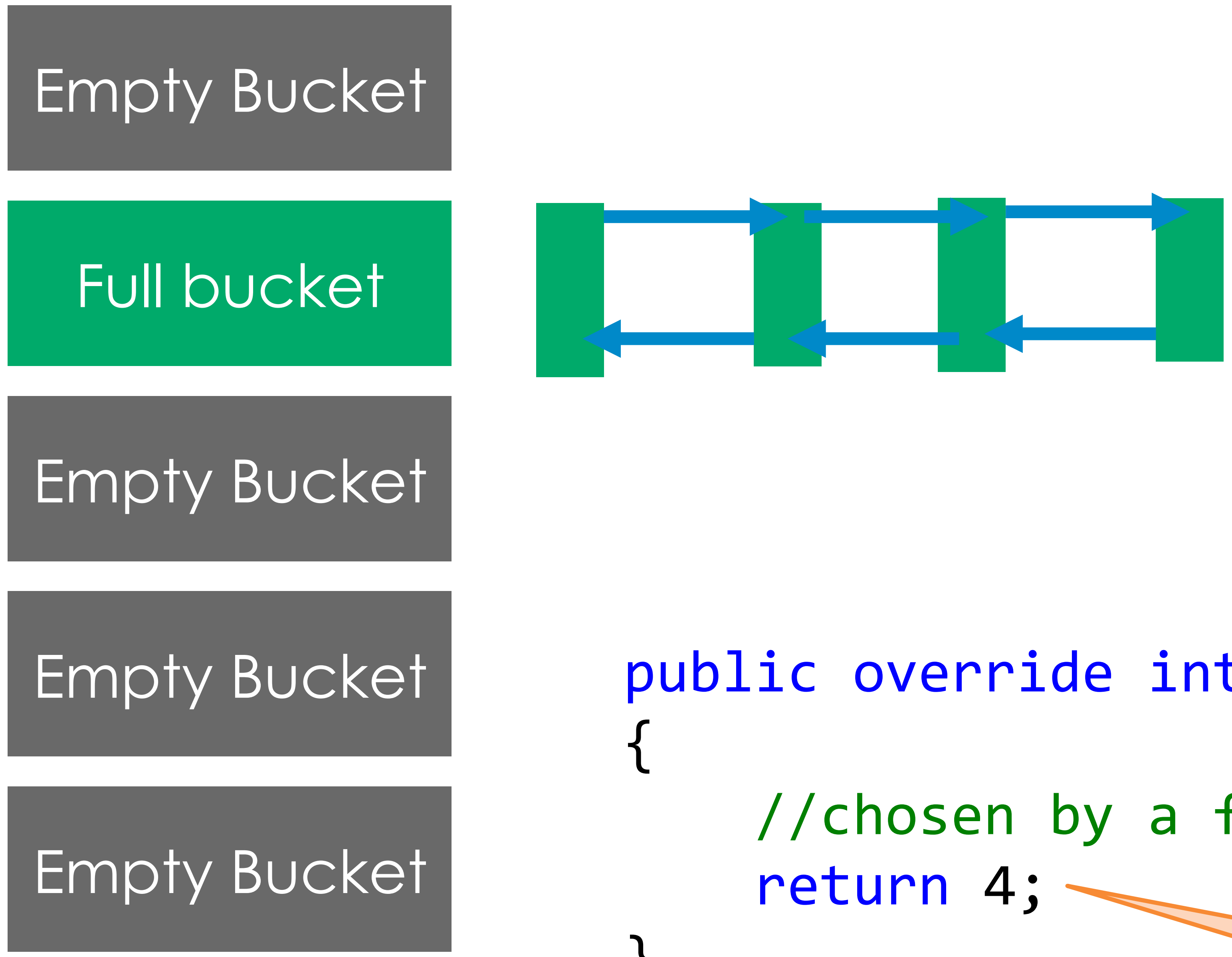
System.Collections.Generic.List



LinkedList



HashSet



1. Call **GetHashCode**
2. Modulus with size
3. Put it or get from to the relevant bucket
4. If hash code changes after put you will never find the item

```
public override int GetHashCode()
{
    //chosen by a fair dice roll!
    return 4;
}
```

Perfectly valid, but will be slow

Property/ Operation	Array	List	LinkedList	Hashset/Dictionary
Index access [key]	🙂	🙂	☹	Not supported/🙂
Add	Not supported	🙂	🙂	🙂
Remove	Not supported	☹	🙂	🙂
Stability	🙂	🙂	🙂	Stable as long as you don't remove ☹
CPU Cache friendliness	🙂	🙂	☹	😕
Contains	☹	☹	☹	🙂

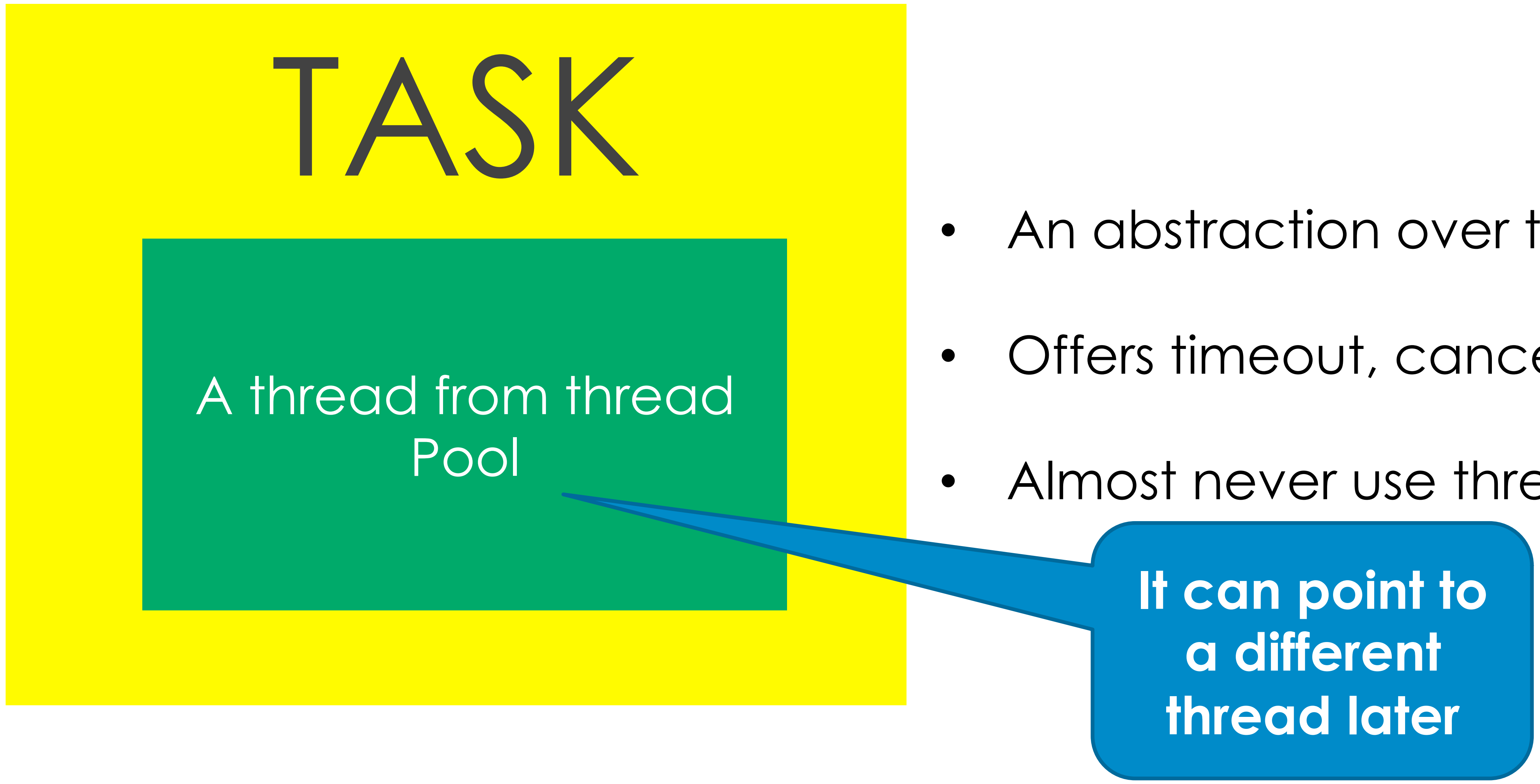
Activity – Collections demo

Section 4.4

Asynchronous programming and tasks

Using .NET Libraries and Runtime

Tasks vs Threads


TASK
A thread from thread Pool
It can point to a different thread later

- An abstraction over threading model
- Offers timeout, cancellation or continuation mechanism
- Almost never use threads, directly but use tasks.

Asynchronous programming and tasks.

- Earlier we were using threads for parallelism.
- Tasks typically wrap and make use of threads but they don’t have to.
- Tasks offer powerful abstraction.
- You can await any task to return the underlying thread to the pool.
- Async await machinery.

Activity - Async and Task Demo

Section 4.5

Monitoring Performance And Usage

Using .NET Libraries and Runtime

Section 4.5

Strings Demo

Using .NET Libraries and Runtime

Assessment Question 1

Which of the following is correct about async and tasks?

A-) async is part of the signature.
B-) await makes the current thread to block and wait.
C-) we can await any task
D-) if an unhandled exception is thrown in a task our application crashes.

Assessment Question 2

Which of the following is correct about Collections?

A-) accessing the 5th element of an array does linear search.
B-) HashSet's contains method does linear search.
C-) We can cast string[] to object[].
D-) We can cast IList<string> to IList<object>.

Lab 2- Fix the bug

In this lesson you learned…

- assemblies and namespaces and how to use them
- different kind of collections like List and Set
- how to debug your applications
- how to monitor performance

Discussion: General Q&A

Lesson 5:

.NET in Practice

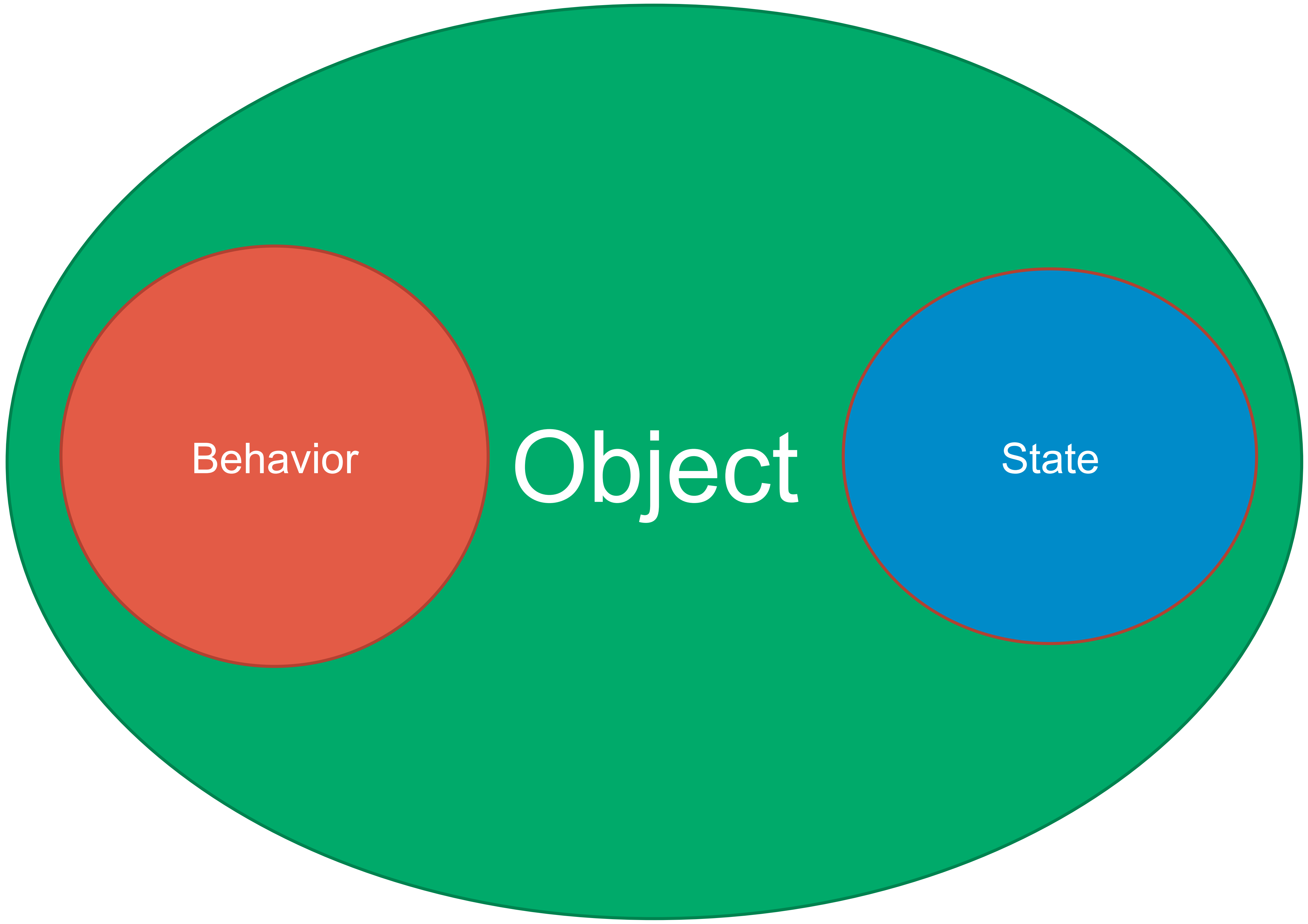
In this lesson you will learn about...

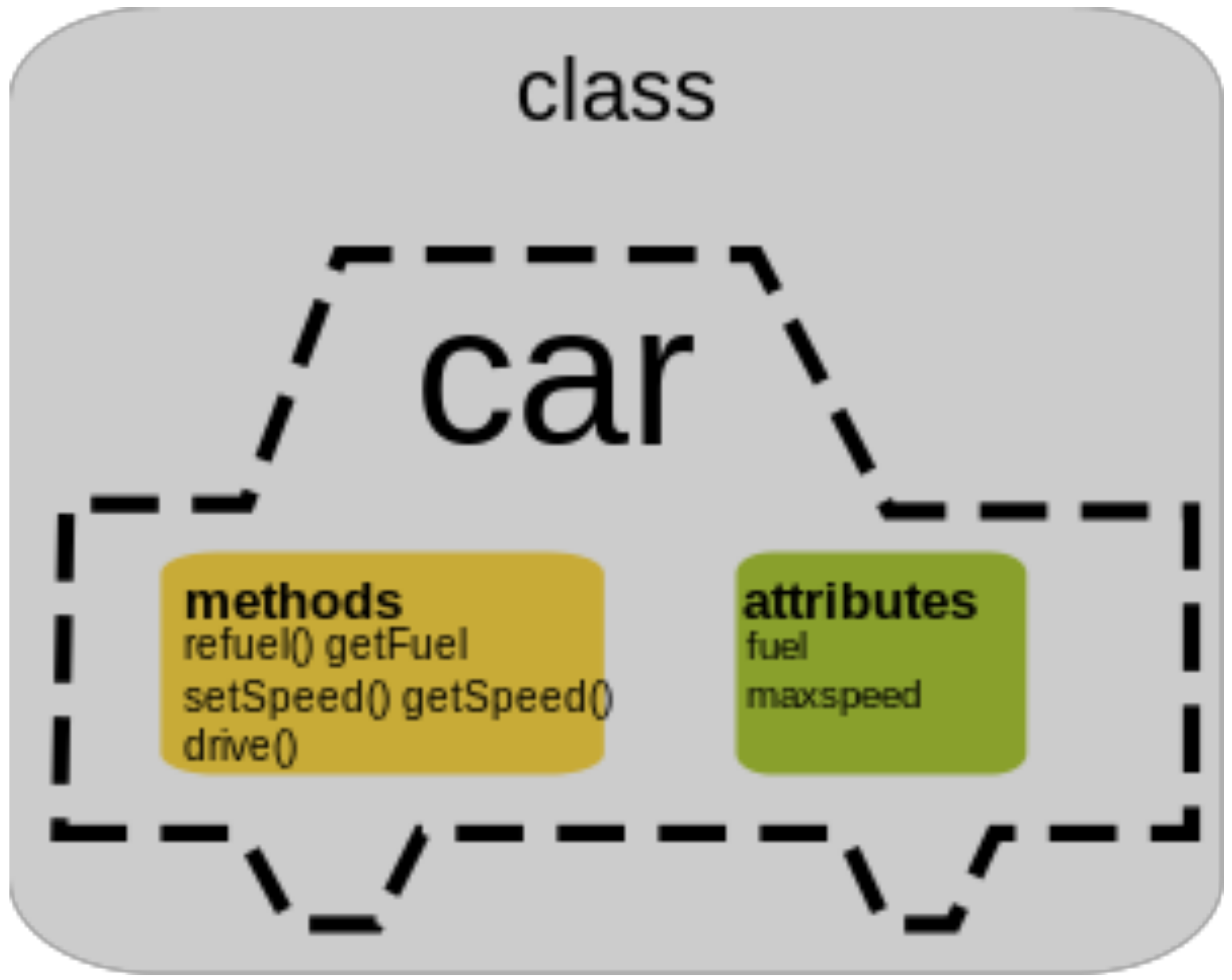
- Understanding OOP
- Implementing Interfaces and Inheriting Classes
- Using reflection

Section 5.1

Understanding OOP

Using .NET Libraries and Runtime


Behavior
Object
State


class
car
methods
refuel() getFuel
setSpeed() getSpeed()
drive()
attributes
fuel
maxspeed

Activity – OOP Demo

Section 5.2

Implementing Interfaces and Inheriting Classes

Asynchronous programming and tasks.

- Inheritance allow us to reuse existing code.
- Polymorphism is achieved by using the virtual keyword.
- Interfaces overcome the limitation of single inheritance.
- Interfaces are contracts.
- Abstract classes are classes that can have not implemented methods and implemented ones together

Activity – Implementing Interfaces Demo

Section 5.3

Reflection

Using .NET Libraries and Runtime

Reflection

- Discover the types and type members at runtime.
- Invoke methods and properties at runtime from their names.

Activity – Reflection Demo

Section 5.4

Generic Host

Using .NET Libraries and Runtime

Assessment Question 1

Which of the following is correct about OOP?

A-) C# supports multiple inheritance.
B-) we can mark a method as sealed even when overriding
C-) An interface can implement one interface at a time.
D-) internal keyword makes a method invisible to other classes.

In this lesson you learned…

- fundemantals of OOP and how to apply them in practice
- How to use reflection
- inheritance and interfaces

Lesson 6:

Storing your data

In this lesson you will learn about...

Relational Database Management Systems

Working with Entity Framework Core

File System and Serialization

Section 6.1

Relational Database Management Systems

Storing your data

Relational databases

- Stores the data in tables.
- Typically you relation two tables by joining them through their foreign keys.
- You use SQL as a language to query the data.
- Mostly supports ACID transactions.
- Mostly supports indexing.
- Sql Server, Oracle, PostgreSQL and MySQL

Activity – Relational Database Demo

Section 6.2

Working with Entity Framework Core

Storing your data

Entity framework

- Entity framework is an Object Relational Mapper.
- Eliminates 90% of SQL code.
- Supports class inheritance.
- Supports database migrations.
- You write your queries in Linq.
- Sql Server supported, Oracle not yet, PostgreSQL and MySQL are also supported

Activity – Entity Framework Demo

Section 6.3

File System and Serialization

Storing your data

File System and Serialization

- Serialization is used to transfer your data from one medium to another. Such as another application, another service, or file system.
- Typically to JSON and XML but binary serializers are available.
- .NET has an extensive file system API.
- Anything IDisposable, don’t forget to dispose it.
- Encoding
- Streams, StreamWriter and StreamReader

Activity – FileSystem and Serializaiton Demo

Assessment Question 1

Which of the following is correct about
Entity Framework?

A-) Entity framework serializes the object to JSON.

B-) Entity framework does not support transactions.

C-) Entity framework syncs our objects to database
when we call SaveChangesAsync

D-)DBContexts are only used for migrations.

Assessment Question 2

Which of the following is correct about
Serialization?
A-) We can serialize object methods.
B-) Sealed types cannot be serialized.
C-) We can deserialize our data to a different class
than the original class
D-) We can’t serialize two objects if they have a reference to each other.

Lab3 – CRUD With EF

Guess My Number Requirements

For our guess my number game,

Save and update the best guess count (minimum number of attempts) as high score

At the beginning of each game show the best score.

In this lesson you learned…

- fundemantals of relational database
- Using entity framework to persist data
- how to use serialization and interact with file system

Lesson 7:

Building mobile apps with Xamarin

In this lesson you will learn about…

- Understanding XAMARIN and XAML
- Using Resources and Templates
- Data Binding
- Animation in Xamarin Forms

Section 7.1

Understanding XAMARIN and XAML

XAMARIN

- Born from mono project allow us to develop cross platform apps.
- Uses XAML as design language.
- Supports iOS, android, UWP, WPF, macOS, GTK#
- Xamarin.Forms : One UI for All!

Activity – XAML Demo

Section 7.2

Databinding

Building mobile apps with Xamarin

Activity – Databinding Demo

Section 7.3

Using Resources and Templates

Building mobile apps with Xamarin

Resources and templates.

- Resources allow you to reuse XAML such as common styles or data to be shared.
- Templates allow you to customize the rendering of data.

Activity – Resources Demo

Activity – Templates Demo

Section 7.4

Animations

Building mobile apps with Xamarin

Animations

- Xamarin offers a simple animation API

Activity – Animation Demo

Assessment Question 1

Which of the following is correct about XAML?
A-) All browsers can render XAML.
B-) We can create instances of regular classes with XAML.
C-) We can use CSS classes with XAML.
D-) We can write C# code inside XAML.

Assessment Question 2

Which of the following is correct about Databinding?
A-) We can bind Commands to Buttons.
B-) Binding is one way only.
C-) We use DataContext for binding in XAMARIN
D-) We use BindingContext for binding in WPF.

In this lesson you learned…

- fundamentals of XAML
- databinding
- animating controls

Lesson 8:

Building Web Applications Using Understanding ASP.NET Core

In this lesson you will learn about...

- Understanding ASP.NET Core
- Using ASP.NET Core MVC Controllers
- Using ASP.NET Core as web service
- C# inside the browser: Blazor

Section 8.1

Understanding ASP.NET Core

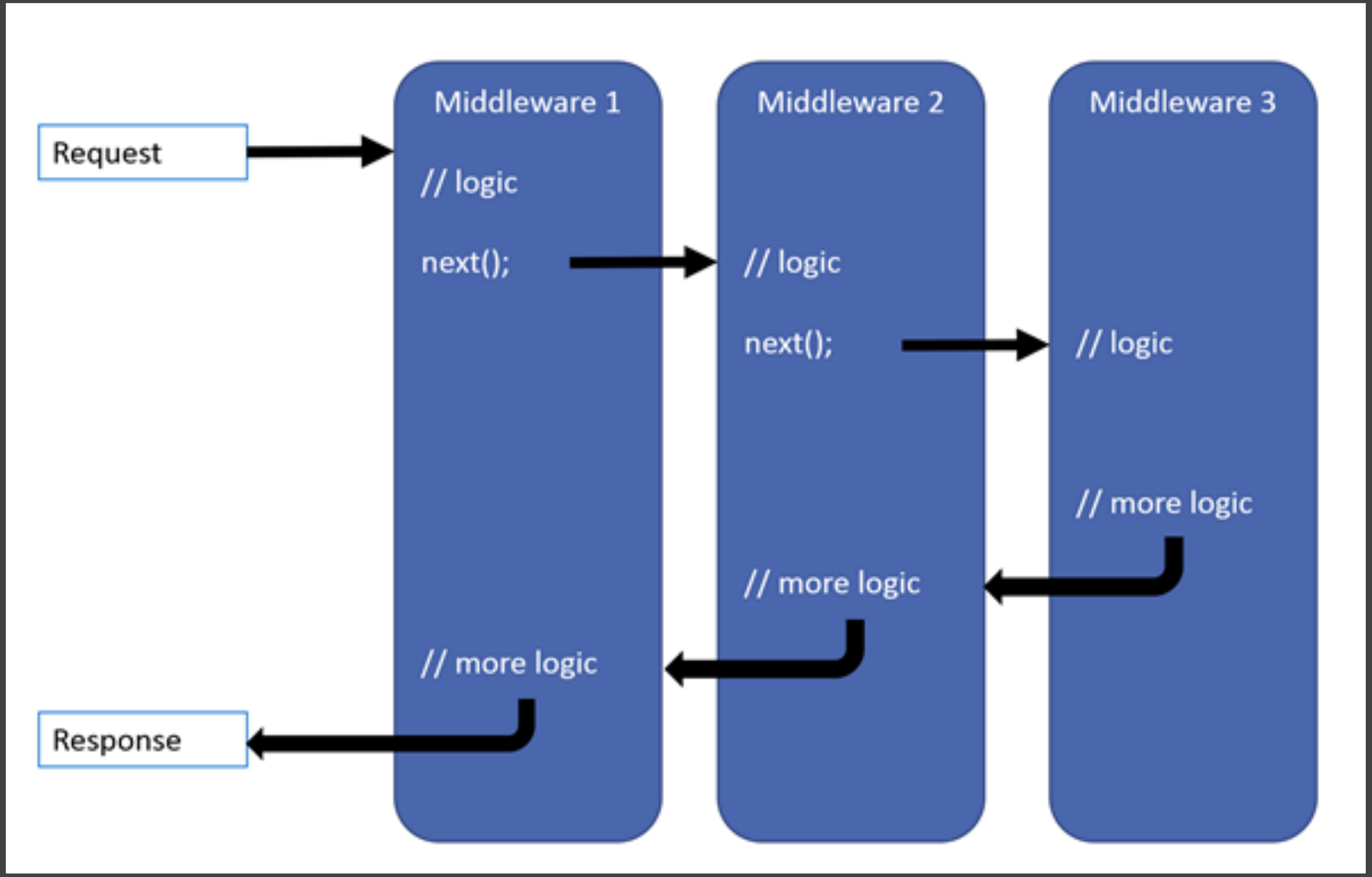
Understanding ASP.NET Core

ASP.NET MVC

```
@model MvcMovie.Models.Movie

@{
    ViewBag.Title = "Edit";
}
<h2>Edit</h2>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    <div class="form-horizontal">
        <h4>Movie</h4>
        <hr />
        @Html.ValidationSummary(true)
        @Html.HiddenFor(model => model.ID)

        <div class="form-group">
            @Html.LabelFor(model => model.Title, new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Title)
                @Html.ValidationMessageFor(model =>
model.Title)
            </div>
        </div>
```

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include="ID,Title,ReleaseDate,Genre,Price")]
Movie movie)
{
    if (ModelState.IsValid)
    {
        db.Entry(movie).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(movie);
}
```

Packt›



ASP.NET Core as a pipeline

ASP.NET Core as a pipeline

```
public class Startup
{
    public void Configure(IApplicationBuilder app)
    {
        app.Use(async (context, next) =>
        {
            // Do work that doesn't write to the Response.
            await next.Invoke();
            // Do logging or other work that doesn't write to the
Response.
        });

        app.Run(async context =>
        {
            await context.Response.WriteAsync("Hello from 2nd
delegate.");
        });
    }
}
```

Section 8.2

Using ASP.NET Core MVC Controllers

Understanding ASP.NET Core

Activity – MVC Demo

Activity – Webservice Demo

Activity –Blazor Demo

Lab 4- Xamarin interop with ASP.NET Core

Guess My Number Requirements

PORT your game to XAMARIN by using Entries, Editors and Buttons. Use ASP.NET Core to save your scores.

Assessment Question 1

Which of the following is correct about
ASP.NET Core?
A-) ASP.NET Core only runs with IIS.
B-) ASP.NET cannot run with IIS.
C-)MVC pattern is mandatory with ASP.NET.
D-)GET method is free of side effects
where as PUT is idempotent.

In this lesson you learned…

- fundamentals of ASP.NET Core
- Middlewares
- MVC Pattern
- Blazor

The next steps

C# Keywords

abstract	as	base	bool
break	byte	case	catch
char	checked	class	const
continue	decimal	default	delegate
do	double	else	enum
event	explicit	extern	false
finally	fixed	float	for
foreach	goto	if	implicit
in	int	interface	internal
is	lock	long	namespace
new	null	object	operator
out	override	params	private
protected	public	readonly	ref
return	sbyte	sealed	short
sizeof	stackalloc	static	string
struct	switch	this	throw
true	try	typeof	uint
ulong	unchecked	unsafe	ushort
using	using static	virtual	void
volatile	while		

C# Keywords

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/

Runtime concepts

- How garbage collector works?
- How JIT works?
- What is an assembly?
- What are value types?
- What is heap and stack?
- What are threads?
- What is boxing?
- What is reflection?

Base Class Library

- What is the difference between Array, List<T>, HashSet<T>?
- What is GetHashCode and how does a Dictionary work?
- What are streams and stream readers writers?
- Serialization to JSON and XML.
- Async IO

Database

- What is an ACID transaction?
- What are transaction isolation levels?
- How are relational databases different than document db?
- How does a database index work ?
- SQL Queries
- BASE transactions?
- CAP theorem?

Design

- Understand why SOLID is append only.
- Understand GOF design patterns are append only.
- Understand the value of testing.
- Understand difference between scalability and performance.
- MVC, MVVM, CQRS and MVU.
- Learn functional programming!

Debug and Diagnose

- Learn proper debugging with Visual Studio
- Remote debugging.
- Code maps
- Intellitrace
- Snapshot debugging

- Learn WinDBG, SOS and Assembly Language
- ILSpy, dnSpy
- Perfview
- Fiddler
- Follow Defrag tools from Channel9
- Know how your Operating System Works. E.G. Working Set, Commit size

Others

- Git: Pull, push,merge and branching and diffing
- A dependency injection framework
- A Logging framework
- An ORM.
- A unit testing framework.
- Typescript, ECMAScript and WebPack
- HTML, CSS
- Vue, Angular and React

Slide10

In this lesson you learned…

- Difference between BDD and TDD
- A practical BDD case.
- An overview of Single Page Applications
- ASP.NET Blazor and WASM.
- Deploying our application to AZURE
- Debugging tricks
- The next steps.

THANK YOU!