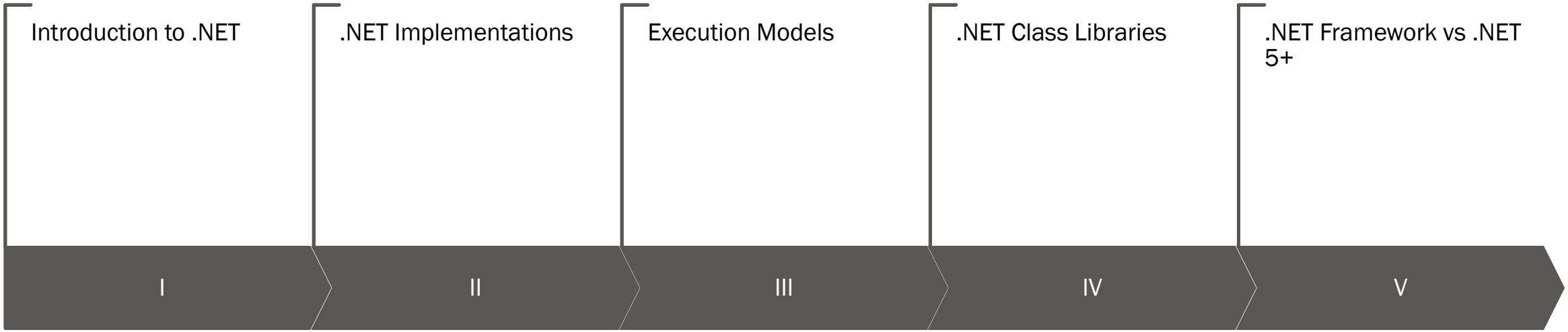
MICROSOFT .NET

ARCTECH INFO

MICROSOFT .NET



INTRODUCTION TO .NET

- .NET is a free, open-source development platform for building many kinds of apps, such as:
  - Web apps, web APIs, and microservices
  - Serverless functions in the cloud
  - Cloud native apps
  - Mobile apps
  - Desktop apps
    - Windows WPF
    - Windows Forms
    - Universal Windows Platform (UWP)
  - Games
  - Internet of Things (IoT)
  - Machine learning
  - Console apps
  - Windows services
- Share functionality among different apps and app types by using class libraries.
- With .NET, your code and project files look and feel the same no matter which type of app you're building.
- You have access to the same runtime, API, and language capabilities with each app.

INTRODUCTION TO .NET (CONTINUED...)

- Cross Platform
- Open Source
- Support

.NET IS CROSS PLATFORM

- You can create .NET apps for many operating systems, including:
  - Windows
  - macOS
  - Linux
  - Android
  - iOS
  - tvOS
  - watchOS
- Supported processor architectures include:
  - x64
  - x86
  - ARM32
  - ARM64
- .NET lets you use platform-specific capabilities, such as operating system APIs.
  - Examples are Windows Forms and WPF on Windows and the native bindings to each mobile platform from Xamarin.

.NET IS OPEN SOURCE

- .NET is open source, using MIT and Apache 2 licenses
- NET is a project of the .NET Foundation
- For more information, see the list of project repositories on GitHub.com

SUPPORT

- .NET is supported by Microsoft on Windows, macOS, and Linux.
- It's updated regularly for security and quality, on the second Tuesday of each month.
- .NET binary distributions from Microsoft are built and tested on Microsoft-maintained servers in Azure and follow Microsoft engineering and security practices.
- Red Hat supports .NET on Red Hat Enterprise Linux (RHEL).
  - Red Hat and Microsoft collaborate to ensure that .NET Core works well on RHEL.
- Tizen supports .NET on Tizen platforms.
  - Tizen is a Linux-based mobile operating system backed by the Linux Foundation (Samsung Electronics)

PROGRAMMING LANGUAGES

- C#
  - C# (pronounced "See Sharp") is a modern, object-oriented, and type-safe programming language. C# has its roots in the C family of languages and will be immediately familiar to C, C++, Java, and JavaScript programmers.
- F#
  - The F# language supports functional, object-oriented, and imperative programming models.
- Visual Basic
  - Among the .NET languages, the syntax of Visual Basic is the closest to ordinary human language, which can make it easier to learn. Unlike C# and F#, for which Microsoft is actively developing new features, the Visual Basic language is stable. Visual Basic isn't supported for web apps, but it is supported for web APIs.

CAPABILITIES OF .NET LANGUAGES

- Type safety
- Type inference - C#, F#, Visual Basic
- Generic types
- Delegates
- Lambdas
- Events
- Exceptions
- Attributes
- Asynchronous code
- Parallel programming
- Code analyzers

IDE

- Visual Studio
  - Runs on Windows only. Has extensive built-in functionality designed to work with .NET. The Community edition is free for students, open-source contributors, and individuals.
- Visual Studio Code
  - Runs on Windows, macOS, and Linux. Free and open source. Extensions are available for working with .NET languages.
- Visual Studio for Mac
  - Runs on macOS only. For developing .NET apps and games for iOS, Android, and web.
- GitHub Codespaces
  - An online Visual Studio Code environment, currently in beta.
- Other 3rd Party IDEs
  - JetBrains Rider

NUGET

- NuGet is an open-source package manager designed for .NET.
- A NuGet package is a .zip file with the .nupkg extension that contains
  - compiled code (DLLs),
  - other files related to that code, and
  - a descriptive manifest that includes information like the package's version number.
- Developers with code to share create packages and publish them to nuget.org or a private host.
- Developers who want to use shared code add a package to their project and can then call the API exposed by the package in their project code.

.NET IMPLEMENTATIONS

- A .NET app is developed for one or more implementations of .NET.
- There are four .NET implementations that Microsoft supports:
  - .NET 5 (and .NET Core) and later versions
    - Latest implementation and runs on any platform (version 1 launched in 2016)
  - .NET Framework
    - Original implementation of .NET (since 2002), and runs only on Windows
  - Mono
    - When a small runtime is required like for mobile development (Xamarin)
  - UWP
    - Build modern windows store apps

EACH IMPLEMENTATION OF .NET INCLUDES THE FOLLOWING COMPONENTS

- One or more runtimes—for example,
  - .NET Framework CLR and
  - .NET 5 CLR.
- A class library—for example,
  - .NET Framework Base Class Library and
  - .NET 5 Base Class Library.
- Optionally, one or more application frameworks are included in .NET Framework and .NET 5+, e.g.
  - ASP.NET,
  - Windows Forms, and
  - Windows Presentation Foundation (WPF)

EXECUTION MODELS

- .NET apps run managed code in a runtime environment known as the Common Language Runtime (CLR).
- The .NET CLR is a cross-platform runtime that includes support for Windows, macOS, and Linux.
- The CLR handles memory allocation and management.
- The CLR is also a virtual machine that
  - not only executes apps
  - but also generates and compiles code using a just-in-time (JIT) compiler.

JIT COMPILER AND IL

- Higher-level .NET languages, such as C#, compile down to a hardware-independent instruction set, called Intermediate Language (IL).
- When an app runs, the JIT compiler translates IL to machine code that the processor understands.
- JIT compilation happens on the same machine that the code is going to run on.
- Since JIT compilation occurs during execution of the application, the compilation time is part of the run time.
- Therefore, JIT compilers have to balance time spent optimizing code against the savings that the resulting code can produce.
- Since JIT compiler knows the actual hardware, it frees developers from having to ship different implementations for different platforms.
- The .NET JIT compiler can do tiered compilation, which means it can recompile individual methods at run time.
- This feature lets it compile quickly while still being able to produce a highly tuned version of the code for frequently used methods.

AUTOMATIC MEMORY MANAGEMENT

- The garbage collector (GC) manages the allocation and release of memory for applications.
- Each time your code creates a new object, the CLR allocates memory for the object from the managed heap.
- As long as address space is available in the managed heap, the runtime continues to allocate space for new objects.
- When not enough free address space remains, the GC checks for objects in the managed heap that are no longer being used by the application. It then reclaims that memory.
- The GC is one of the CLR services that help ensure memory safety.
- A program is memory safe if it accesses only allocated memory.
- For instance, the runtime ensures that an app doesn't access unallocated memory beyond the bounds of an array.

WORKING WITH UNMANAGED RESOURCES

- Sometimes code needs to reference unmanaged resources.
- Unmanaged resources are resources that aren't automatically maintained by the .NET runtime.
- For example, a file handle is an unmanaged resource.
- A FileStream object is a managed object, but it references a file handle, which is unmanaged.
- When you're done using the FileStream, you need to explicitly release the file handle.
- In .NET, objects that reference unmanaged resources implement the IDisposable interface.
- When you're done using the object, you call the object's Dispose() method, which is responsible for releasing any unmanaged resources.
- The .NET languages provide a convenient using statement (C#, F#, VB) that ensures the Dispose method is called.

RUNTIME LIBRARIES

- .NET has an expansive standard set of class libraries, known as runtime libraries, framework libraries, or the base class library (BCL).
- These libraries provide implementations for many general-purpose and workload-specific types and utility functionality.
- There are three types of class libraries that you can use:
  - Platform-specific class libraries have access to all the APIs in a given platform (for example, .NET Framework on Windows, Xamarin iOS), but can only be used by apps and libraries that target that platform.
  - Portable class libraries have access to a subset of APIs, and can be used by apps and libraries that target multiple platforms.
  - .NET Standard class libraries are a merger of the platform-specific and portable library concept into a single model that provides the best of both.

.NET FRAMEWORK VS .NET 5+

- .NET Framework
  - Launched in 2002
  - The original .NET Implementation
  - Contains legacy code
  - The current version 4.8 is the last version and there will be no more new versions of .NET Framework planned in the future.
  - .NET Framework run-times can be installed side by side with other implementations

- .NET 5+ (originally called .NET Core)
  - Launched In 2016
  - The modern .NET implementation
  - Was built from the ground up
  - The current version is .NET 6.0 and the recommended implementation for new projects.
  - .NET 5+ run-times can be installed side by side with other implementations.