ASP.NET CORE 2.0 – Part 1

Thien Nguyen, Apr-2018
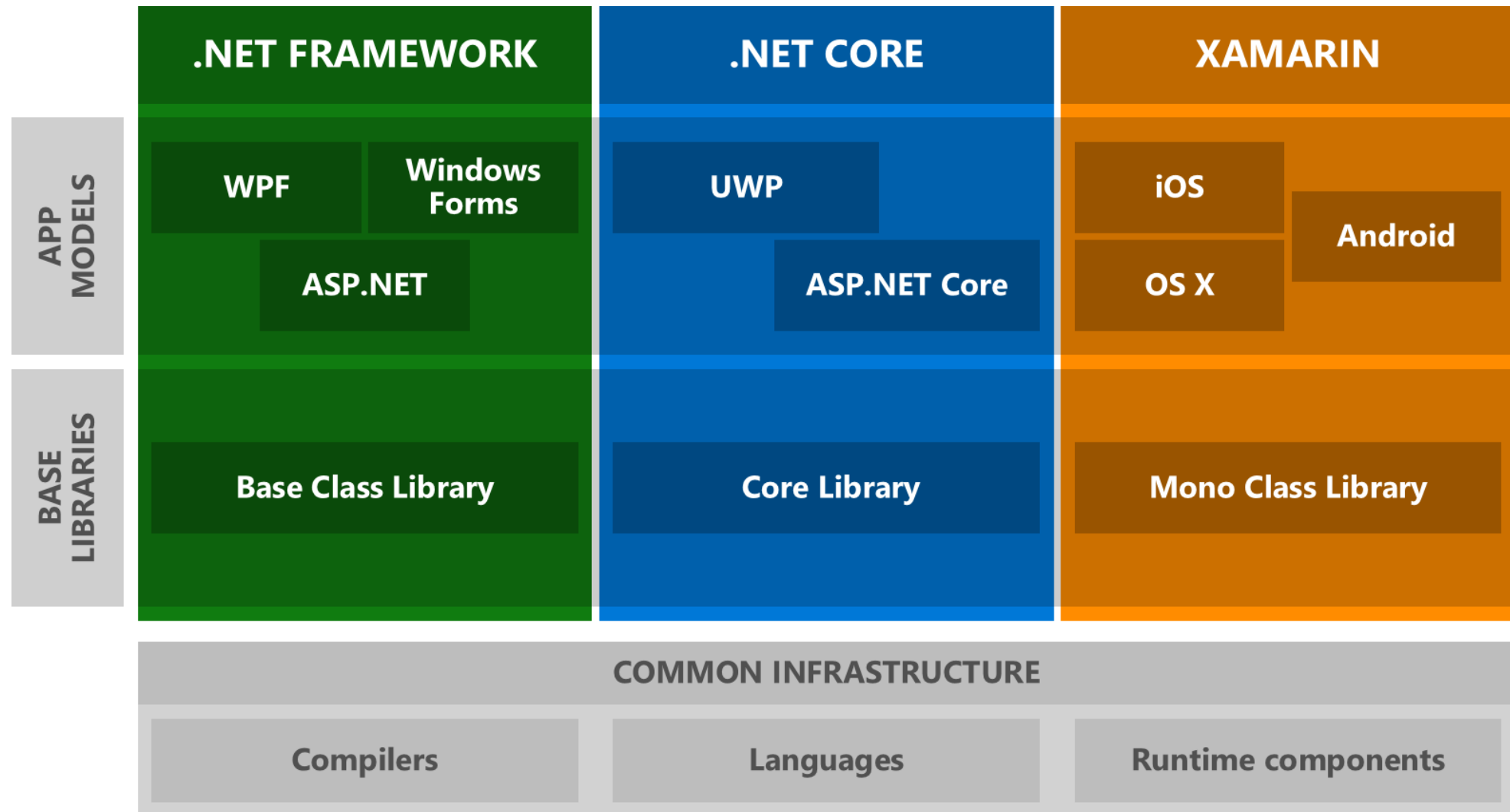
Agenda

- Introduction

- .NET Core Overview

- .NET Standard

- ASP.NET Core Fundamentals

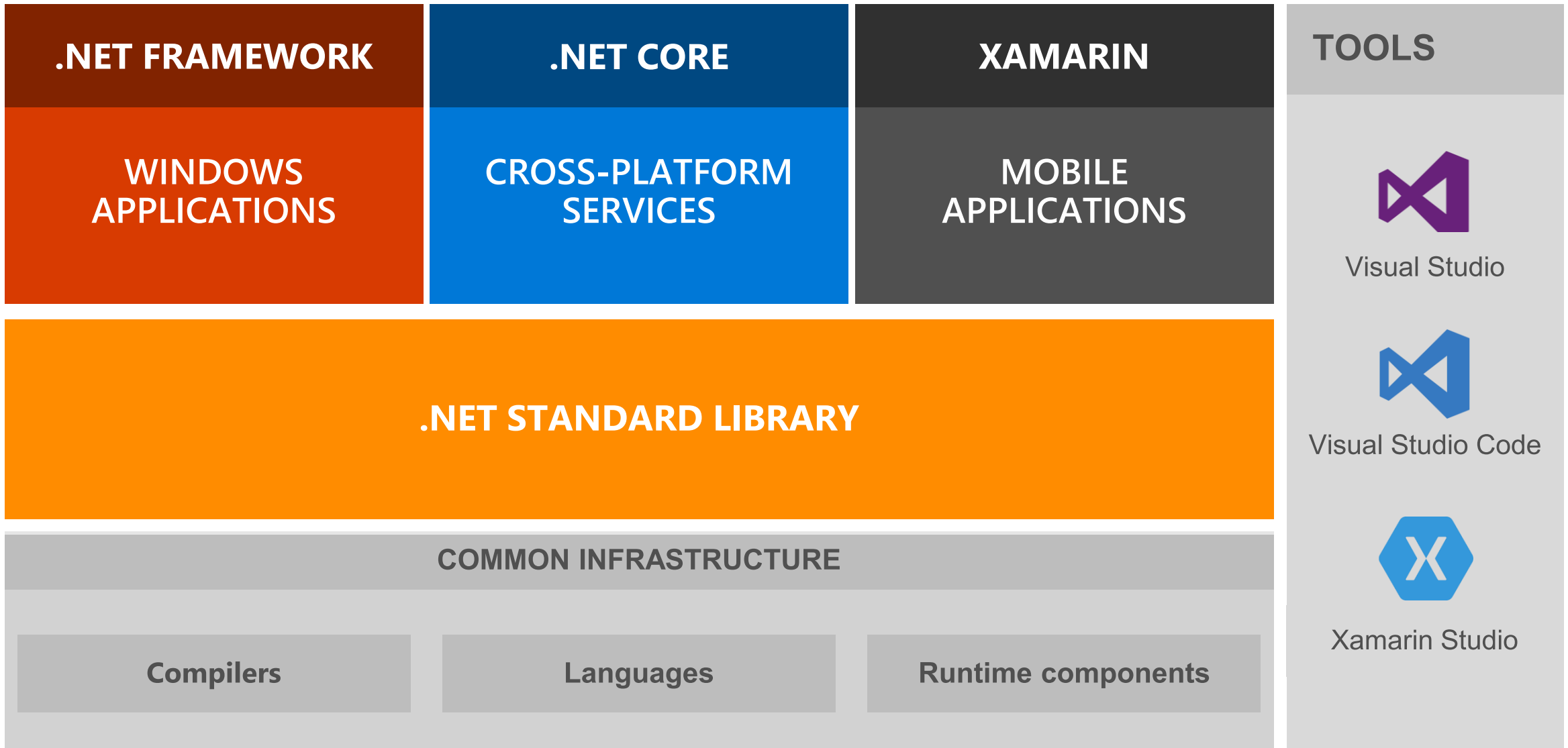- What news in ASP.NET Core MVC

- Demo

- Q&A

Problems of .NET

- Windows only

- Closed

- All or nothing monolithic framework

- 15 years old

.NET Yesterday

Unified platform

.NET FRAMEWORK	.NET CORE	XAMARIN
WINDOWS APPLICATIONS	CROSS-PLATFORM SERVICES	MOBILE APPLICATIONS

.NET STANDARD LIBRARY

COMMON INFRASTRUCTURE

Compilers	Languages	Runtime components

TOOLS

Visual Studio

Visual Studio Code

Xamarin Studio

.NET Standard

.NET Standard allows sharing code, binaries, and skills between .NET client, server, and all flavors

- .NET Standard provides a specification for any platform to implement

- All .NET runtimes provided by Microsoft implement the standard

.NET Standard

.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0
.NET Framework	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1	4.6.1	4.6.1
Mono	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.14
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.8
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	8.0
UWP	10.0	10.0	10.0	10.0	10.0	10.0.16299	10.0.16299	10.0.16299
Windows	8.0	8.0	8.1					
Windows Phone	8.1	8.1	8.1					
Windows Phone Silverlight	8.0							

.NET Standard 2.0

Has much bigger API surface

Extended to cover intersection between .NET Framework and Xamarin

Also makes .NET Core 2.0 bigger as it implements .NET Standard 2.0

+20K

More APIs than
.NET Standard 1.x

Can reference .NET Framework libraries

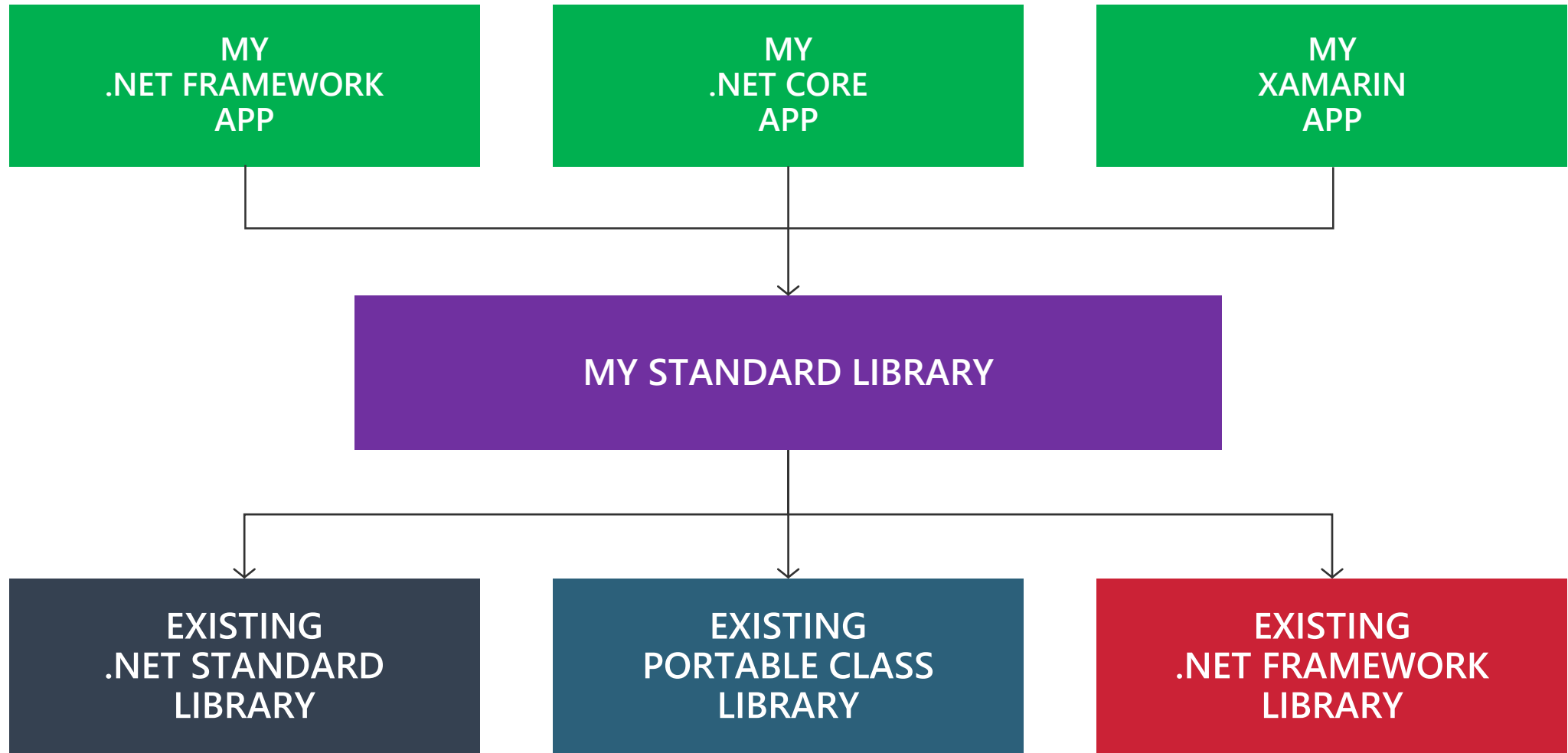Compatibility shim allows referencing existing .NET Framework binaries

No recompile required – also covers existing NuGet packages

Limited to libraries that only use APIs that are available for .NET Standard

~70%

of NuGet packages
are API compatible

.NET Standard Bridges

.NET Core is the Future

- A general purpose development platform

- Cross-platform, supporting Windows, macOS and Linux

- Can be used in device, cloud, and embedded/IoT.

.NET Core Components

- ## A .NET Runtime – CoreCLR
  Provides a type system, assembly loading, a garbage collector, native interop and other basic services

- ## A set of Framework Libraries – CoreFX
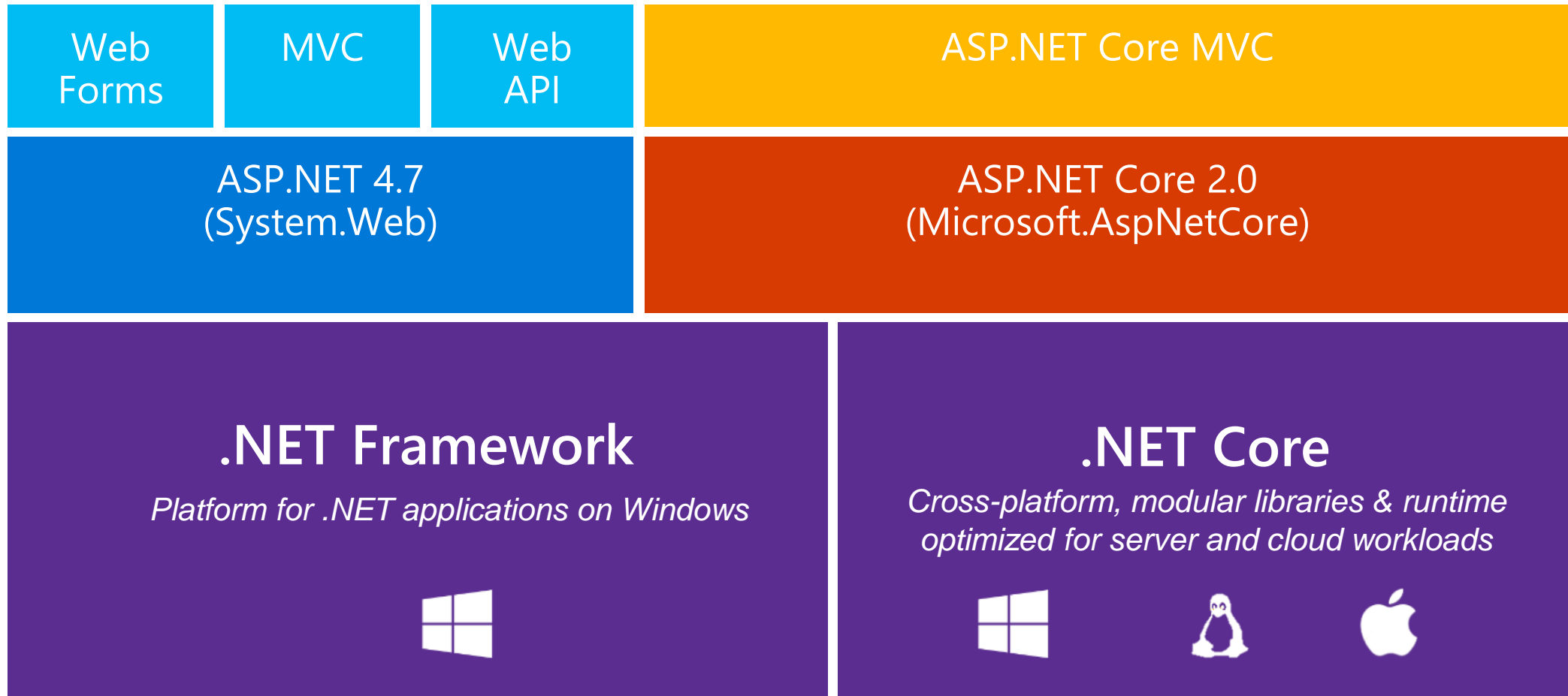 Provide primitive data types, fundamental utilities

- ## .NET Core SDK
  CLIs, Tools, Language compilers

- ## The 'dotnet' app host
  To launch .NET Core apps

ASP.NET CORE in Nutshell

Web Forms	MVC	Web API	ASP.NET Core MVC

ASP.NET 4.7 (System.Web)	ASP.NET Core 2.0 (Microsoft.AspNetCore)

.NET Framework
Platform for .NET applications on Windows

.NET Core
Cross-platform, modular libraries & runtime optimized for server and cloud workloads

ASP.NET Core

Cross-platform

Windows, Linux and MacOS.

Fast

A top performing web framework on TechEmpower benchmarks

Lightweight

No impact deployment and a modular development model perfect for containers

Innovation

Built-in dependency injection, new configuration model, Middlewares, TagHelpers, ViewComponents, etc.
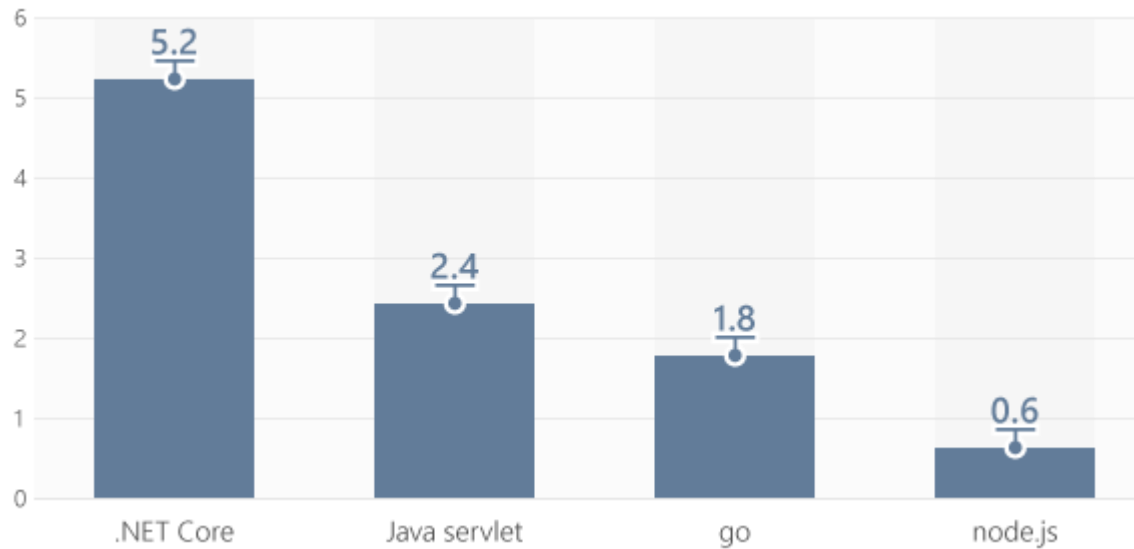
Open source

Runtime, libraries, compiler, languages and tools developed in the open in GitHub

Kestrel

- Kestrel is a cross-platform web server based on libuv, a cross-platform asynchronous I/O library

  - Fast

  - Not a fully featured web server

  - Recommended to be run behind a proxy

More on performance :https://www.techempower.com/benchmarks/

ASP.NET vs ASP.NET Core – Main Differences

ASP.NET

- IIS, Windows only

- System.Web, Included all by default

- HTTP Modules, HTTP Handlers

- MVC + Web API + Web Pages

- Web.config

ASP.NET CORE

- Kestrel, Cross-platform

- No System.Web, Everything is Nuget packages. There is no dll by default

- Middlewares

- ASP.NET Core MVC

- .json, .ini, environment variables, .etc

ASP.NET vs ASP.NET Core – New Features

- Build-in dependency injection

- Built-in logging API and providers

- TagHelpers

- ViewComponents

- Application Part

- …

Middleware
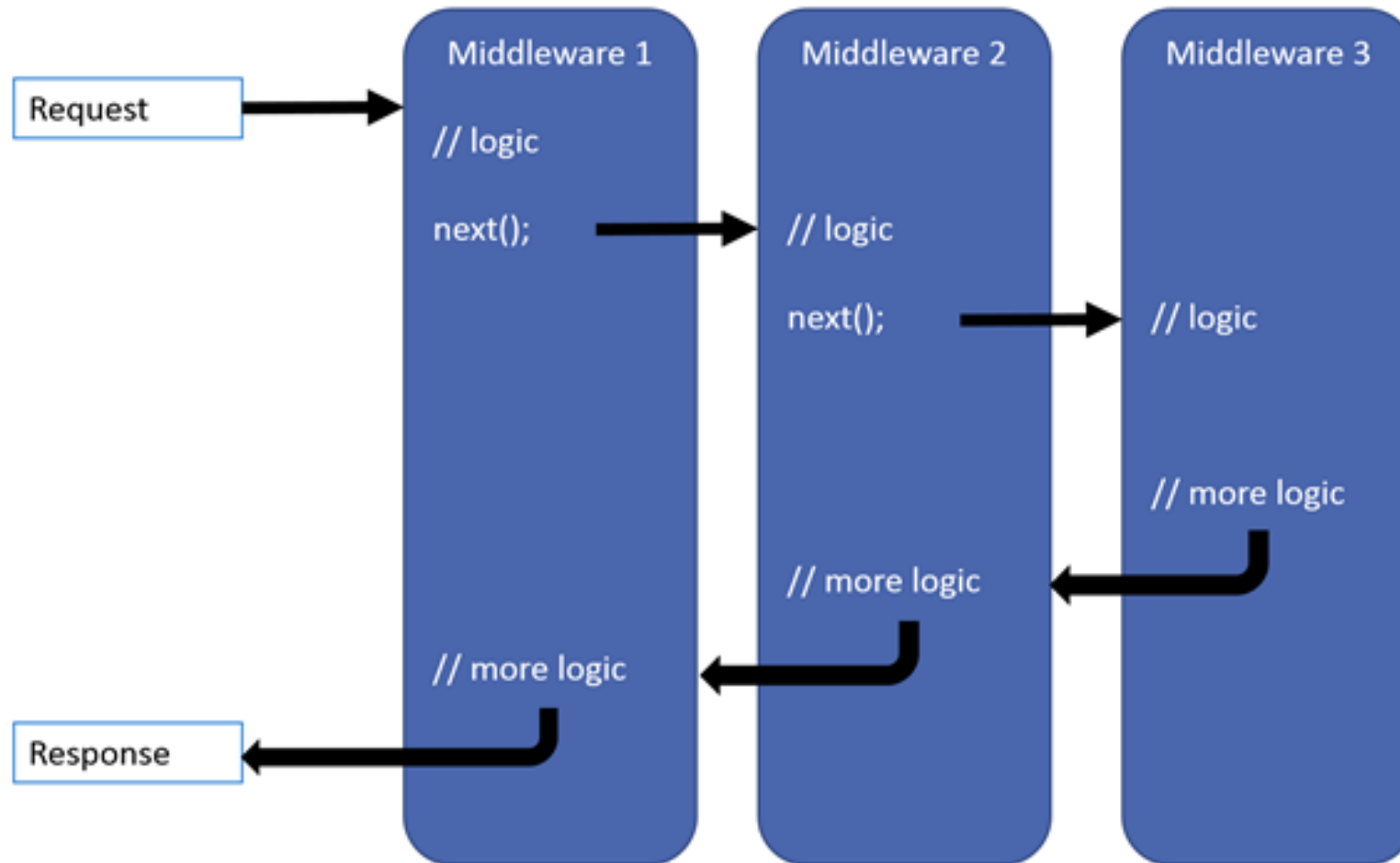
Middlewares

- Routing
- Authentication
- Static files
- Diagnostics
- Error handling

- Session
- CORS
- Localization
- Custom

Demo

- Create an ASP.NET Core project

- Project structure

- Write simple middlewares

- Working with CLI

- .deps.json file

Dependency Injection

- ASP.NET Core is designed to support dependency injection

- Default service container

- Can integrated with other IOC container: Autofac, Unity, StrutureMap, etc.

- Can Inject to Razor

View component

- There is no child action in MVC Core

- How they are different?

    - No model binding

    - No action filters

    - Not reachable from HTTP

    - ViewBag/ViewData are shared with controller

Deployment
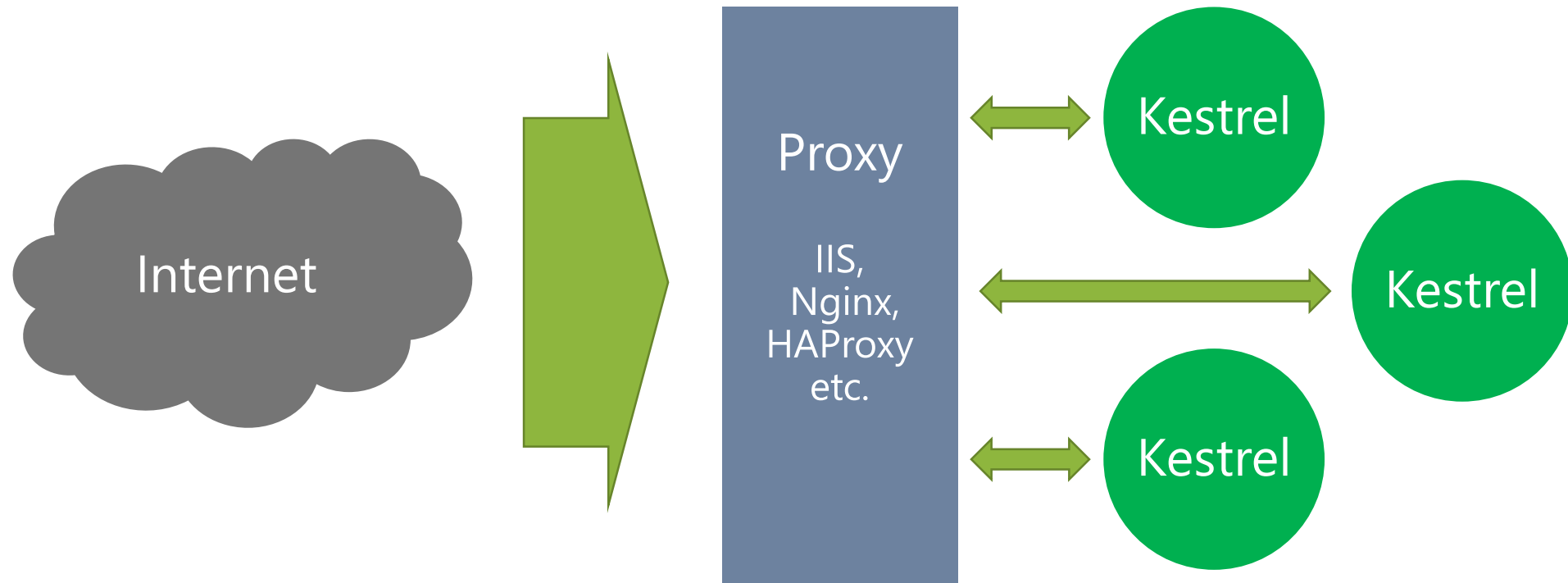
Deployment Options

Self Contained

- App carries everything with it
  .NET Core doesn't need to be pre-installed

- Runs like a normal native executable
  Built for specific platform runtime (Windows, Linux distro, etc.)

- Bigger size

Framework Dependent

- App contains only its own code
  .Net Core need to be pre-installed in target system

- Run by dotnet cli
  Portable between installations of .NET Core

- Smaller size

Q&A

THANK YOU

www.nashtechglobal.com