I will make you a better C# developer 2018 edition .NET Standard (short version)

Kathleen Dollard

Principal Program Manager, Microsoft

kdollard@microsoft.com

https://www.youtube.com/channel/UCaFP8iQMTuPXinXBMEXsSuw

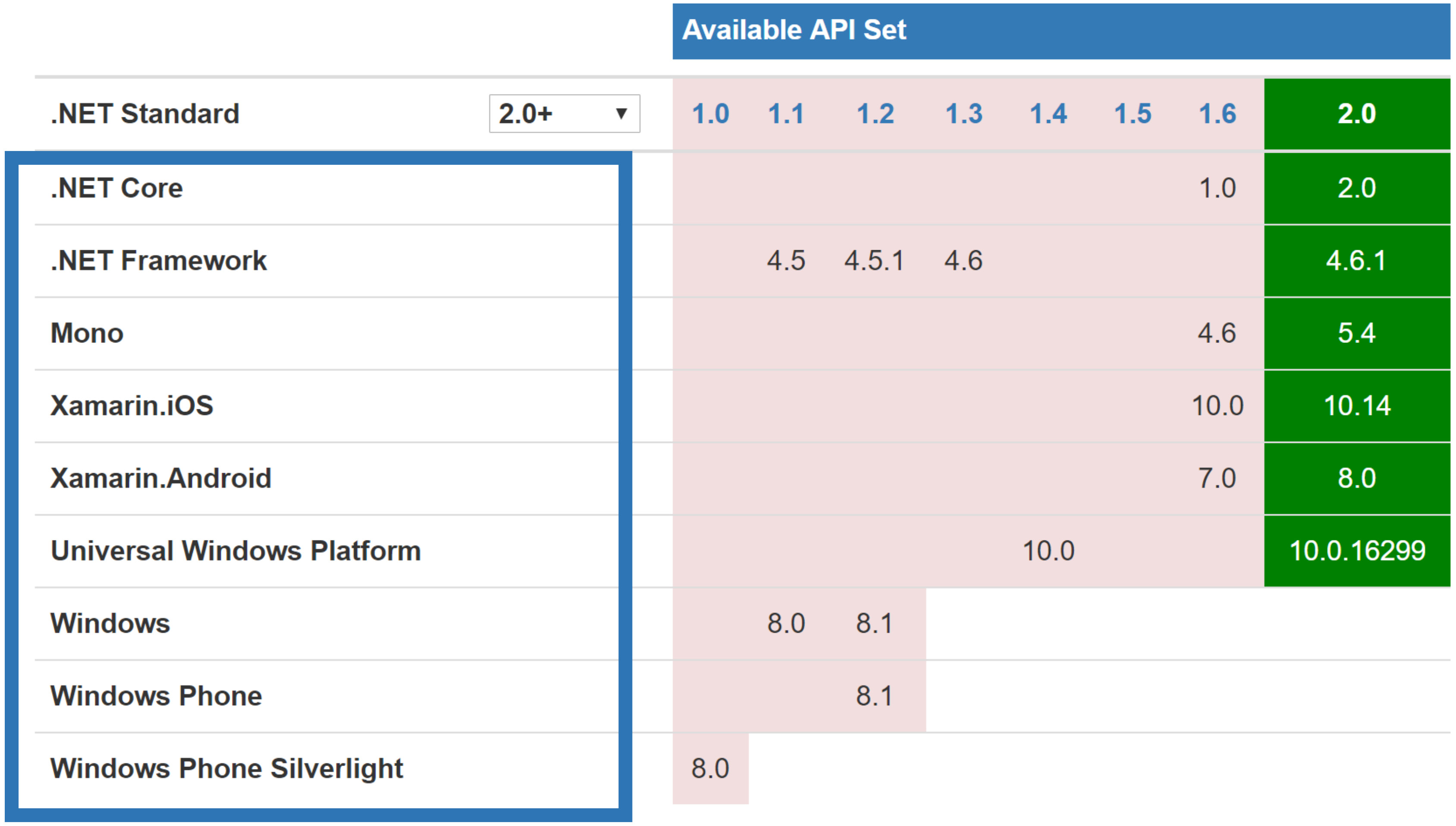
Search:

YouTube Immo .net

- .NET Full Framework
  - The .NET framework we've been using for years and love
  - .NET 4.5.2…NET 4.7.1 (Fall Creator's Update)
- .NET Core
  - A new cross platform .NET
  - Doesn't do Windows stuff
  - .NET Core 2.0 with .NET Core 2.1 in preview
- .NET Standard
  - A ***standard*** for .NET
  - Definition for what makes something able to be .NET
  - .NET Standard 2.0, no 2.1 preview

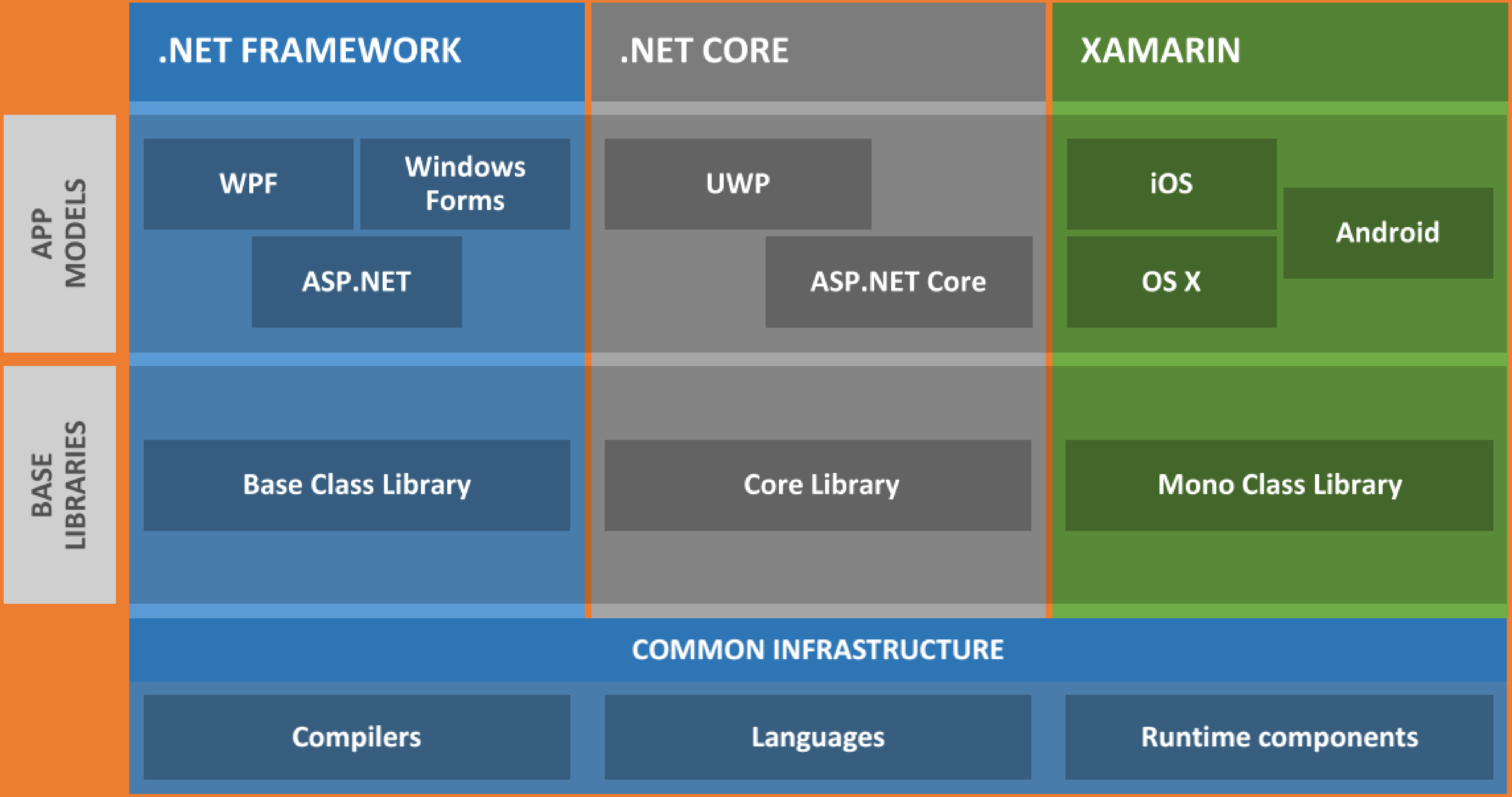
- .NET Full Framework ***implementation***
  - The .NET framework we've been using for years and love
  - .NET 4.5.2….NET 4.7.1 (Fall Creator's Update)
- .NET Core ***implementation***
  - A new cross platform .NET that doesn't do Windows stuff
  - .NET Core 2.0 with .NET Core 2.1 in preview
- .NET Standard ***standard or specification***
  - A ***standard*** for .NET
  - Definition for what makes something able to be .NET
  - .NET Standard 2.0, no 2.1 preview

Other implementations

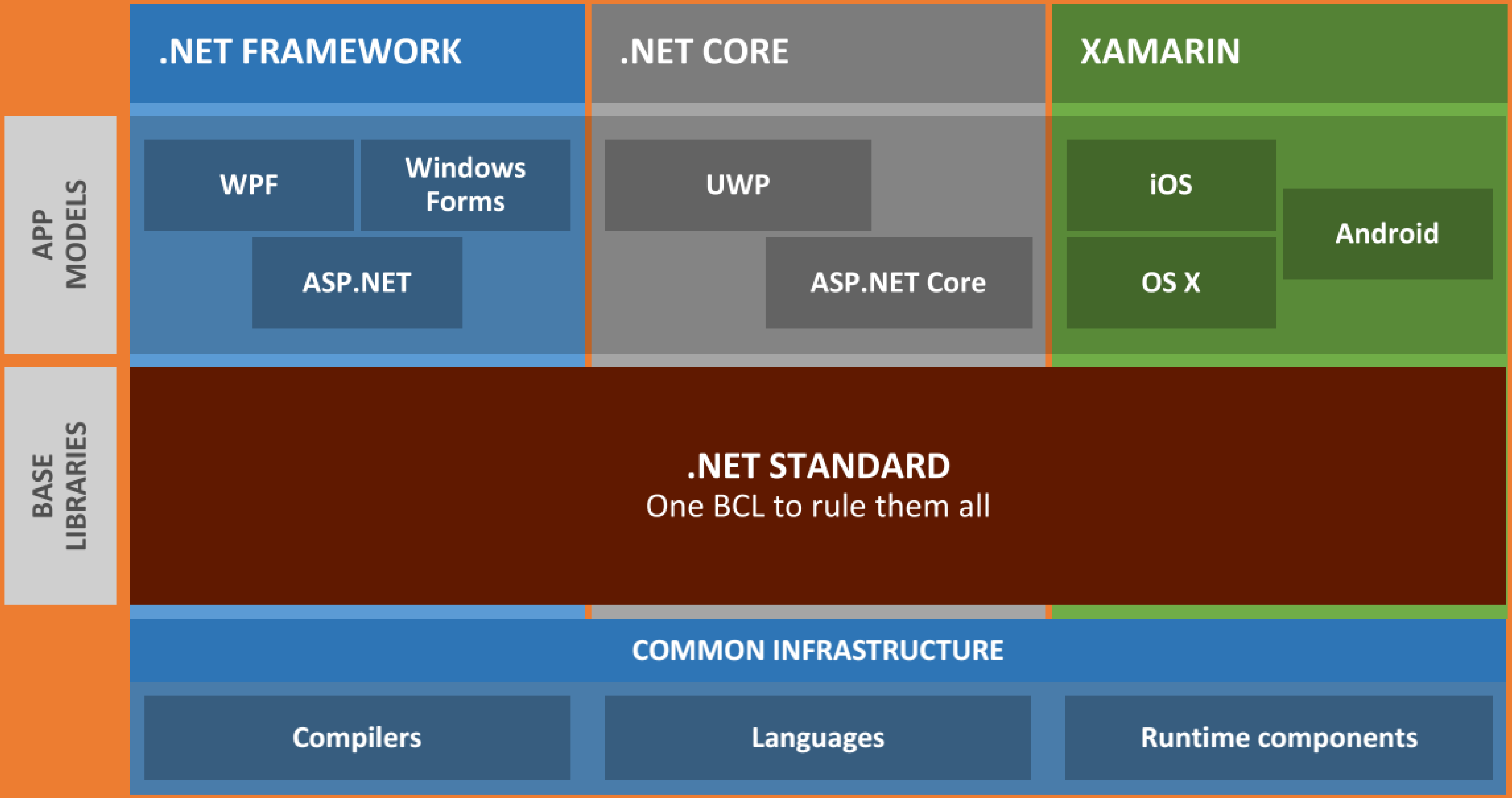


	Available API Set							
.NET Standard 2.0+	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core							1.0	2.0
.NET Framework		4.5	4.5.1	4.6				4.6.1
Mono							4.6	5.4
Xamarin.iOS							10.0	10.14
Xamarin.Android							7.0	8.0
Universal Windows Platform					10.0			10.0.16299
Windows		8.0	8.1					
Windows Phone			8.1					
Windows Phone Silverlight	8.0							

Without .NET Standard

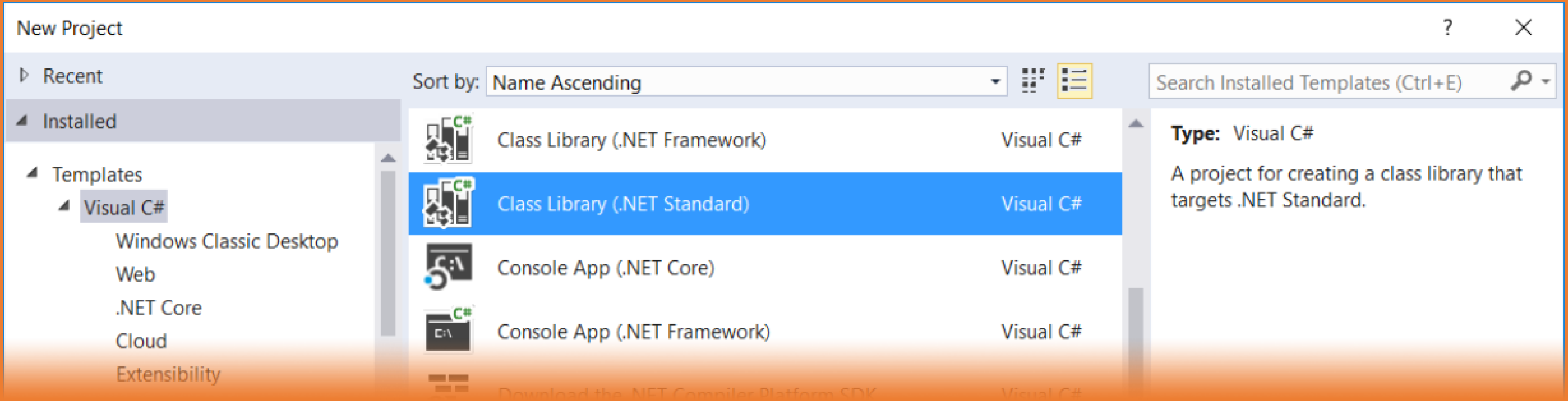


With .NET Standard



What is .NET Standard

- It is a specification
- It represents a set of APIs all .NET Standard platforms implement
- If you're class libraries just use Standard, they conform to it
  - .NET Core supplies a host letting you can .NET Standard libraries

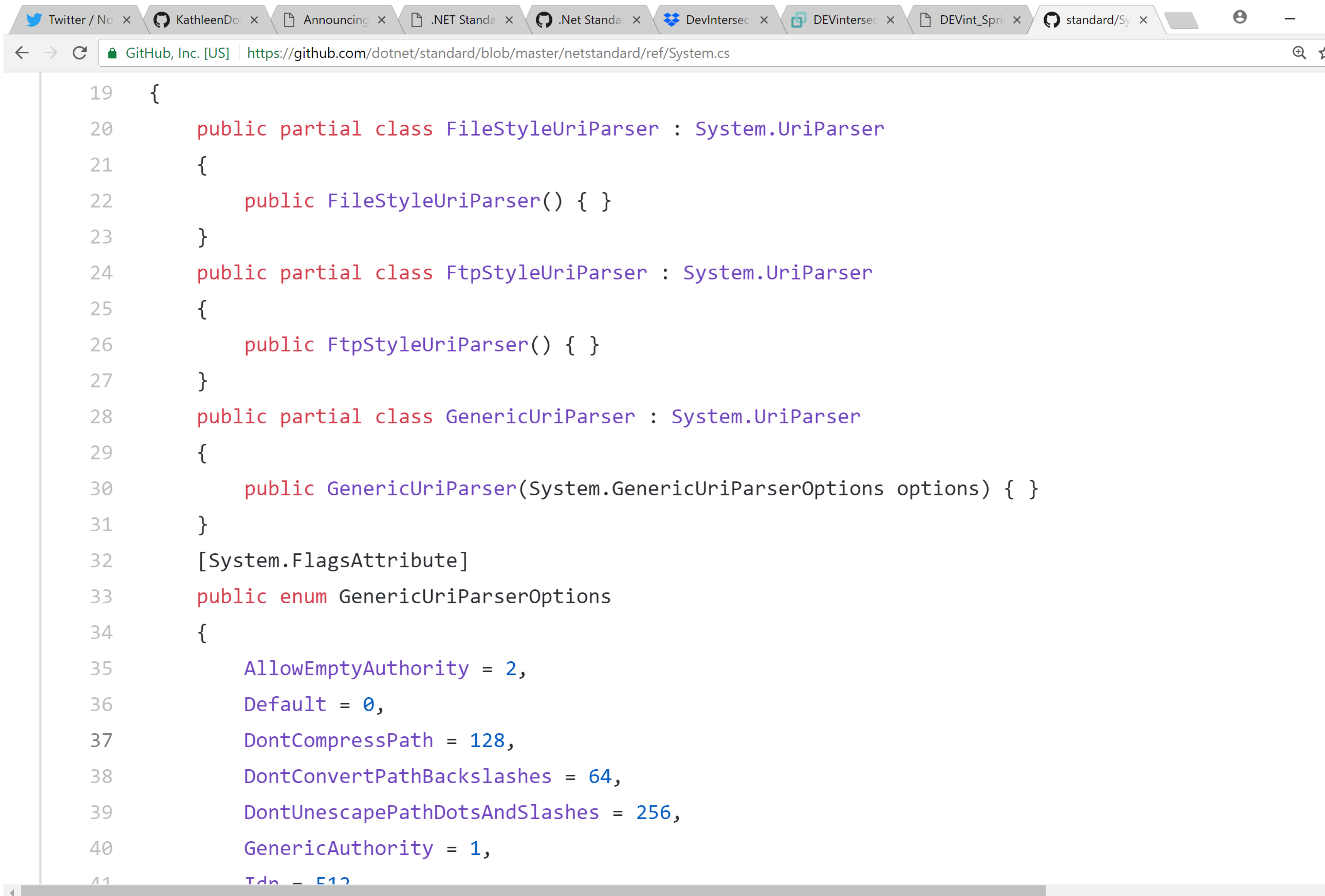


Class Library (.NET Framework)	Visual C#
Class Library (.NET Standard)	Visual C#
Console App (.NET Core)	Visual C#
Console App (.NET Framework)	Visual C#
Download the .NET Compiler Platform SDK	Visual C#

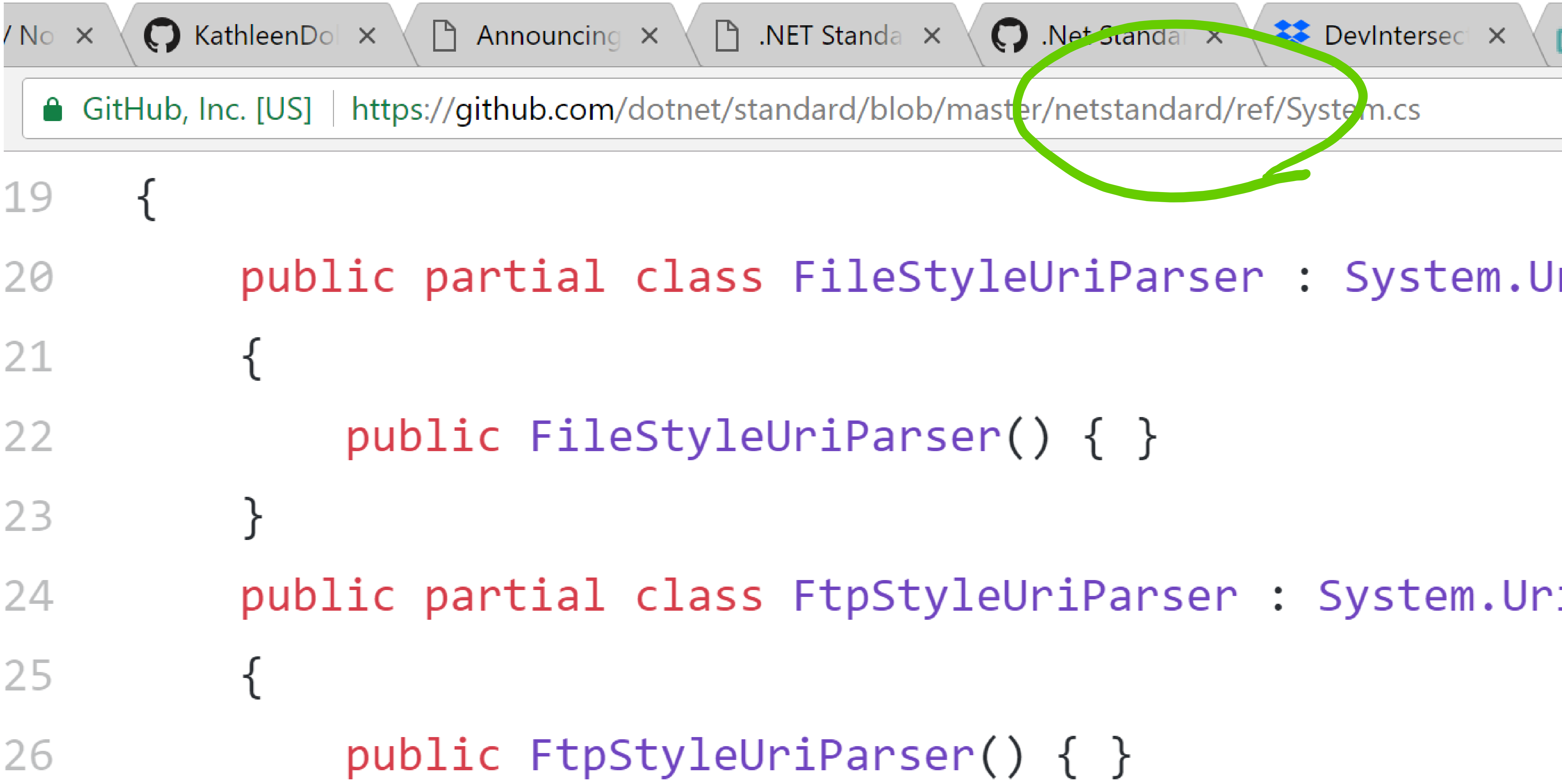
Type: Visual C#

A project for creating a class library that targets .NET Standard.

So .NET Standard is a standard, a document written in code



```
{
    public partial class FileStyleUriParser : System.UriParser
    {
        public FileStyleUriParser() { }
    }
    public partial class FtpStyleUriParser : System.UriParser
    {
        public FtpStyleUriParser() { }
    }
    public partial class GenericUriParser : System.UriParser
    {
        public GenericUriParser(System.GenericUriParserOptions options) { }
    }
    [System.FlagsAttribute]
    public enum GenericUriParserOptions
    {
        AllowEmptyAuthority = 2,
        Default = 0,
        DontCompressPath = 128,
        DontConvertPathBackslashes = 64,
        DontUnescapePathDotsAndSlashes = 256,
        GenericAuthority = 1,
        Idn = 512
```



https://github.com/dotnet/standard/blob/master/netstandard/ref/System.cs

```
{
    public partial class FileStyleUriParser : System.Ur
    {
        public FileStyleUriParser() { }
    }
    public partial class FtpStyleUriParser : System.Ur
    {
        public FtpStyleUriParser() { }
```

Wat! ***BINARY*** compat

How does .NET Standard work?

.NET Standard is represented by

- The NuGet package **NetStandard.Library** which contains
- The reference assembly **netstandard.dll**

At build time

- .NET Standard bridges references to existing .NET Framework and PCL assemblies via type forwarding

At runtime

- Each platform provides an implementation for netstandard.dll that type forwards to its implementation

I thought Portable Class Libraries were supposed to do handle platforms?

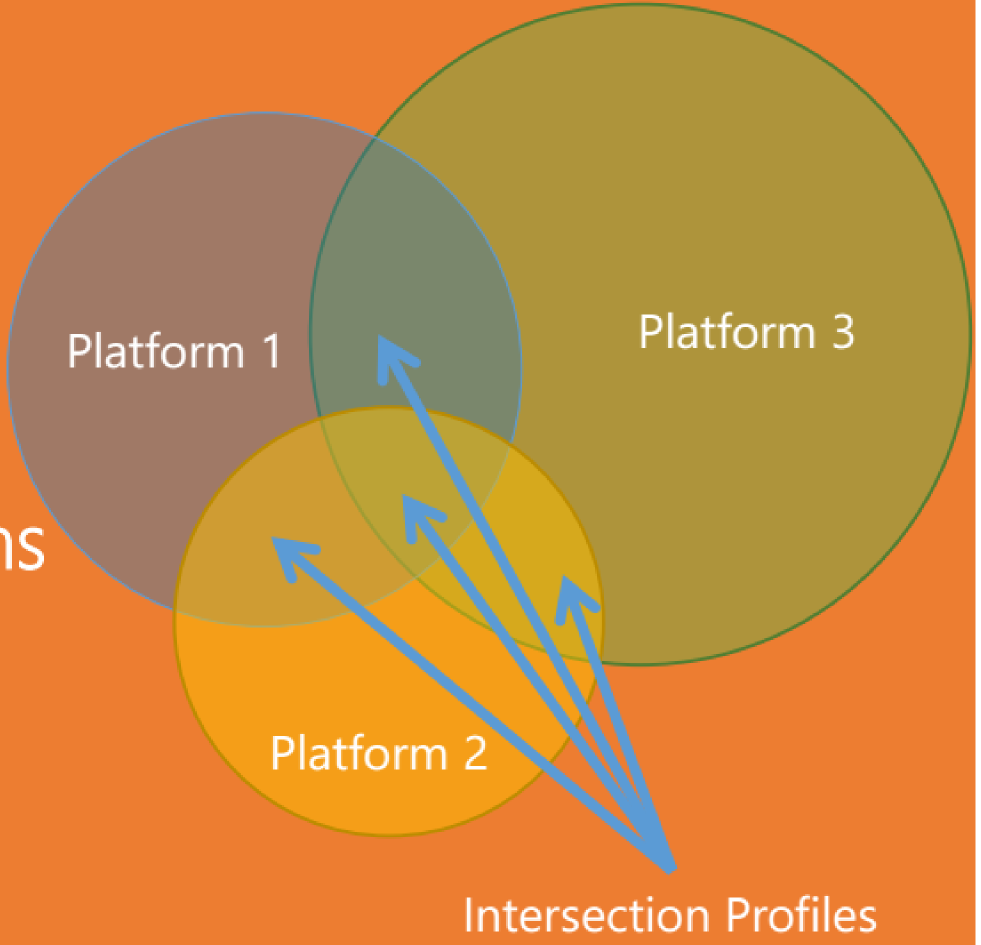
Difference to Portable Class Libraries (PCL)

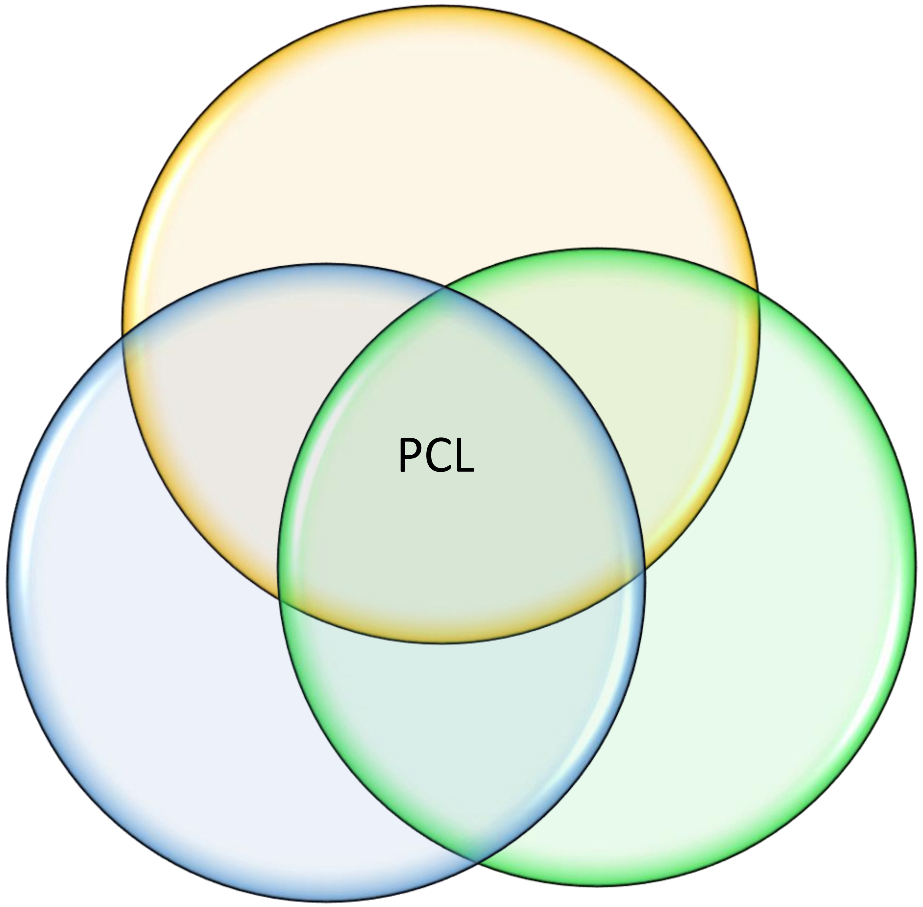
PCLs were an **after thought**, i.e. each **platform could decide** which APIs to includes

- No systematic approach to versioning
- Computed intersection profiles

Each PCLs is targeting a **specific set of platforms**

- Not compatible with newer platforms
- Hard to understand compatibility relationships



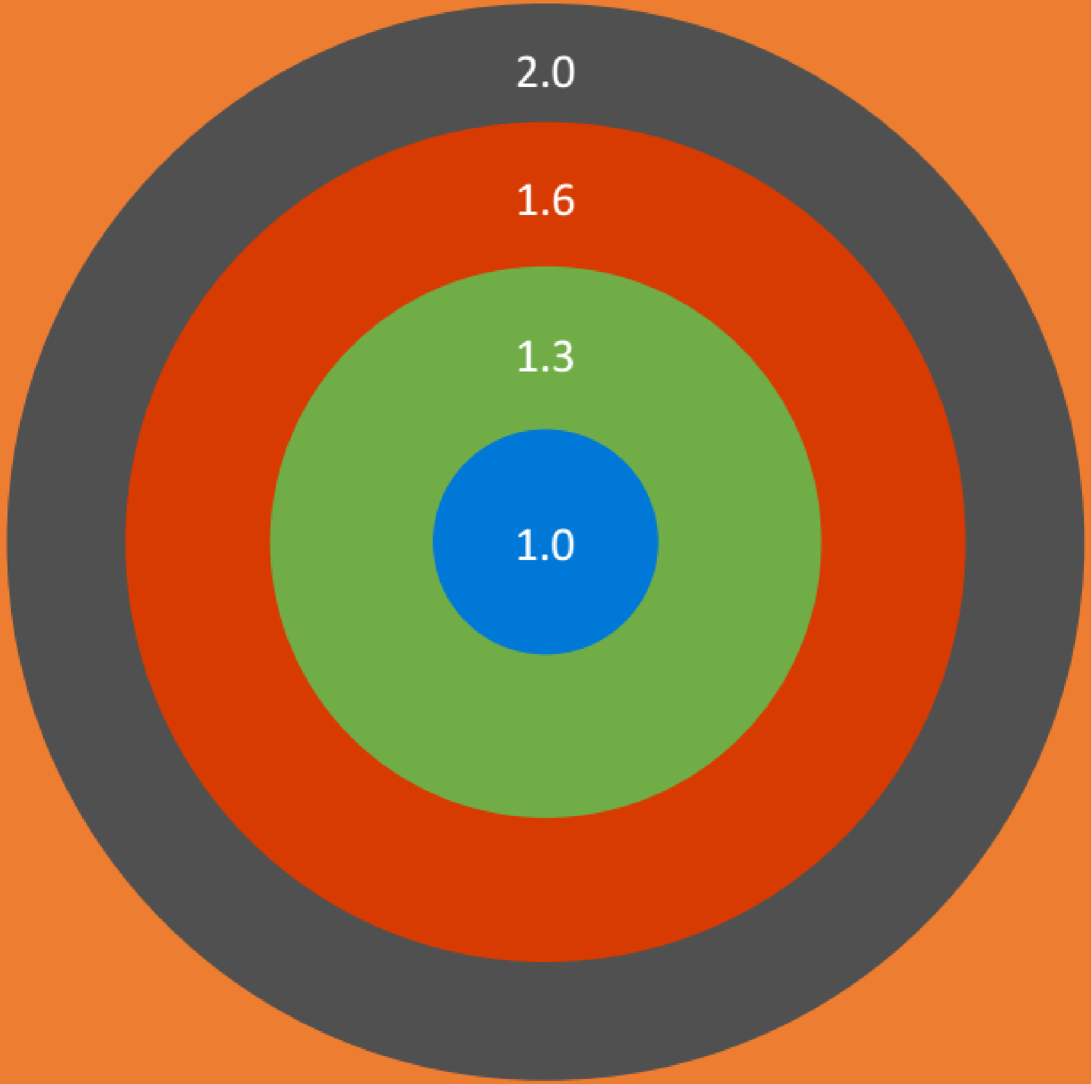


PCL is driven by other libraries

.NET Standard drives

And now we are going to chat about versions

How does versioning work in .NET Standard?

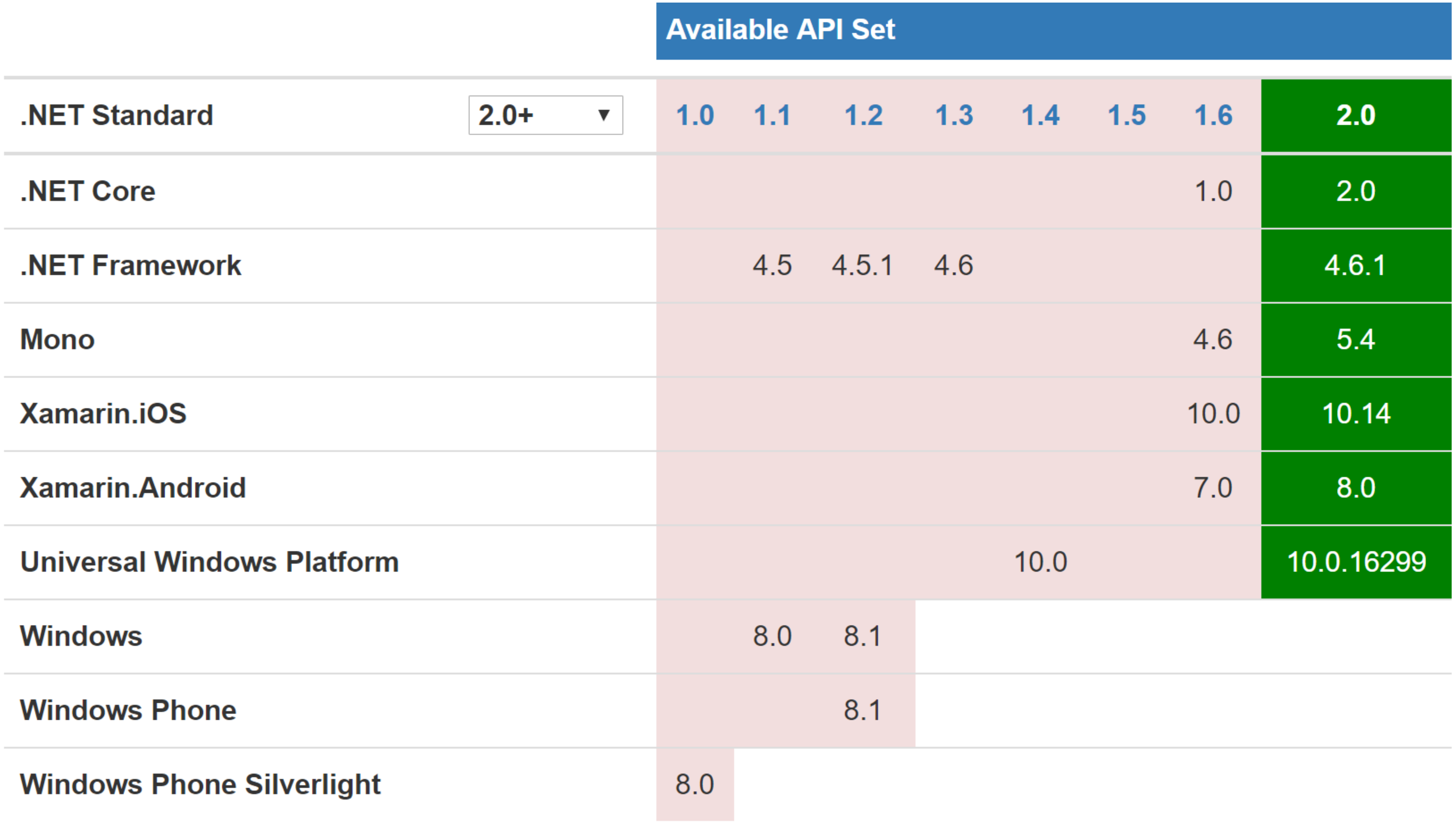


Higher versions incorporate all APIs from previous versions.

- Projects targeting version *X.Y* can reference libraries & projects targeting any version between 1.0 and X.Y

Concrete .NET platforms implement a specific version of .NET Standard

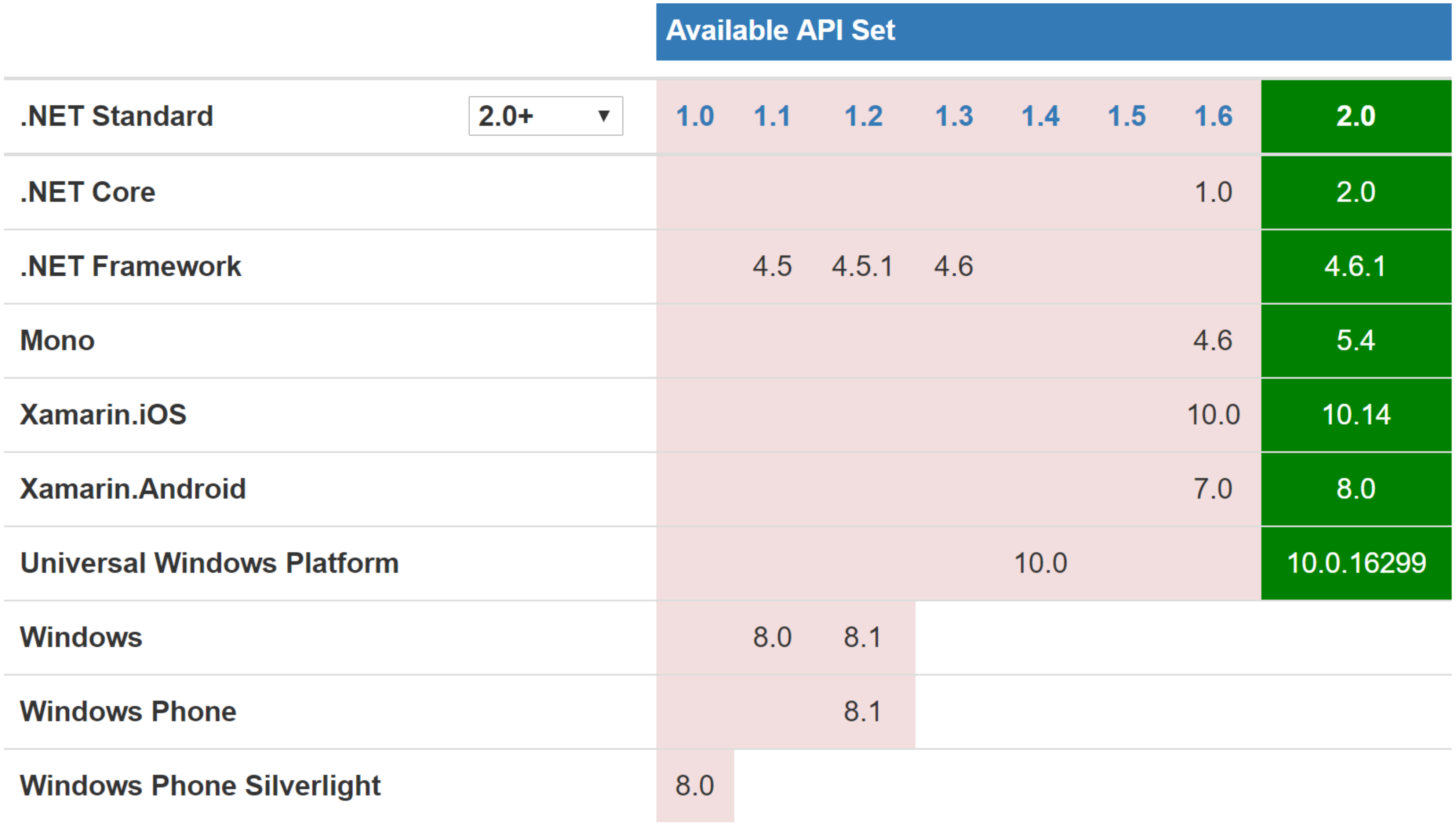
- From that platform you can reference libraries up to that version



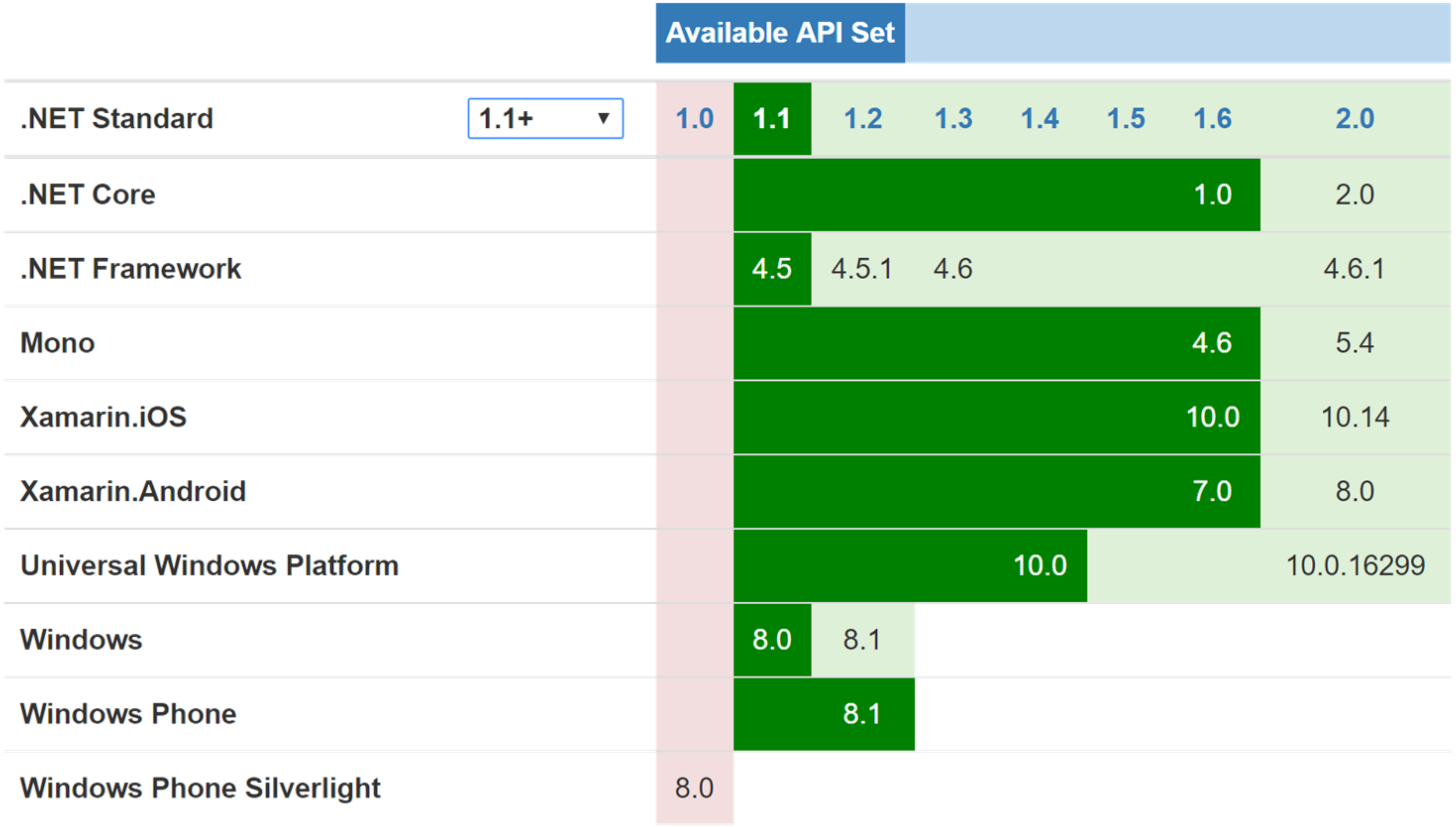
		Available API Set							
.NET Standard	2.0+ ▾	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core								1.0	2.0
.NET Framework			4.5	4.5.1	4.6				4.6.1
Mono								4.6	5.4
Xamarin.iOS								10.0	10.14
Xamarin.Android								7.0	8.0
Universal Windows Platform						10.0			10.0.16299
Windows			8.0	8.1					
Windows Phone				8.1					
Windows Phone Silverlight		8.0							

Deciding on a Version

- Create a standard library to work with an existing app targeting 4.5.0
    - Find .NET 4.5.1 in table-that’s the version of .NET Standard you need to use
- Create a green-field app with .NET Standard architecture
    - Use .NET Standard 2.0
- Create a library for broadest possible reach
    - Use the lowest version of .NET Standard that has the features you require



	Available API Set							
.NET Standard 2.0+	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core							1.0	2.0
.NET Framework		4.5	4.5.1	4.6				4.6.1
Mono							4.6	5.4
Xamarin.iOS							10.0	10.14
Xamarin.Android							7.0	8.0
Universal Windows Platform					10.0			10.0.16299
Windows		8.0	8.1					
Windows Phone			8.1					
Windows Phone Silverlight	8.0							



	Available API Set							
.NET Standard 1.1+ ▾	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core							1.0	2.0
.NET Framework		4.5	4.5.1	4.6				4.6.1
Mono							4.6	5.4
Xamarin.iOS							10.0	10.14
Xamarin.Android							7.0	8.0
Universal Windows Platform					10.0			10.0.16299
Windows		8.0	8.1					
Windows Phone			8.1					
Windows Phone Silverlight	8.0							

The Version Table

To get the static version table

- https://github.com/dotnet/standard/blob/master/docs/versions.md

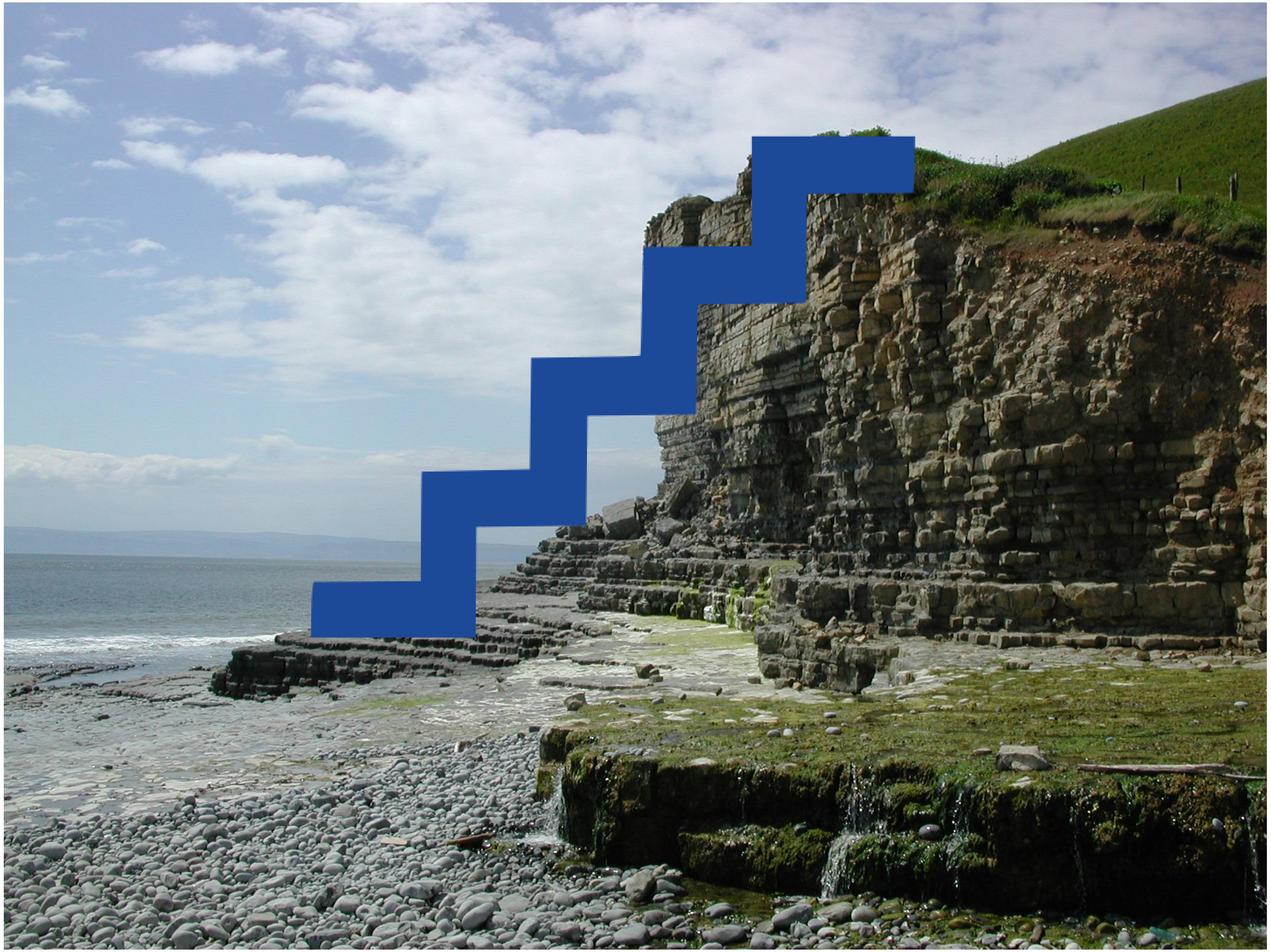
To get the dynamic version table

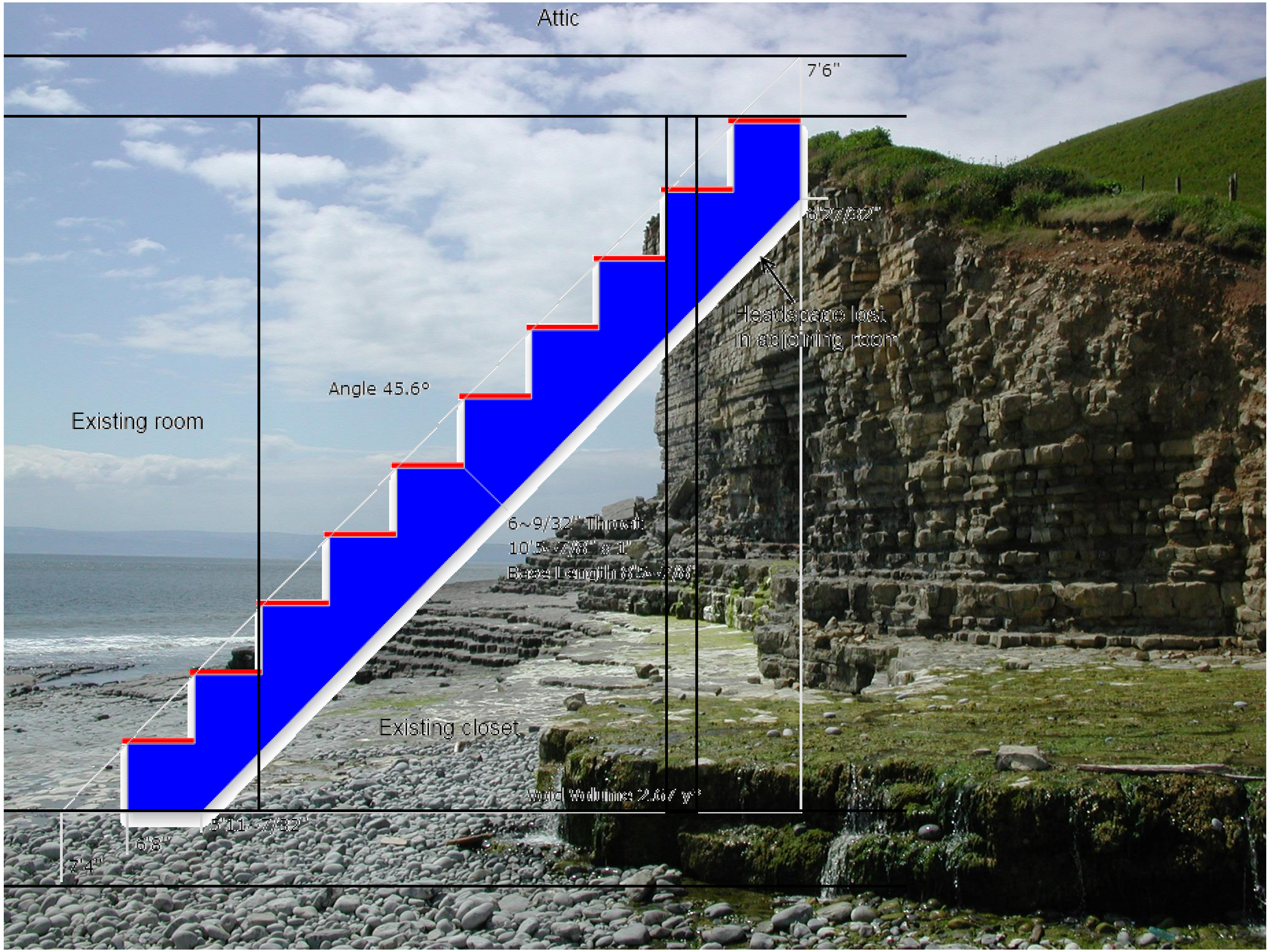
- http://immo.landwerth.net/netstandard-versions/#

What about the next version of Standard?

- It’s a reference assembly, it only evolves on new features
  - Don’t expect to see patches
- Platforms must adapt for a new .NET Standard to be meaningful
  - Imagine a new column in the table
  - Initially empty
  - As each platform adapts, it gets an entry
- Don’t worry about seeing another massive change soon
  - Platforms and Standard now largely align

And what about me?




Attic
7'6"
6'27/32"
Headspace lost
in adjoining room
Angle 45.6°
Existing room
6~9/32" Throat
10'5~7/8" x 1"
Base Length 8'5~7/8"
Existing closet
Void Volume 2.67 y³
5'11~7/32"
6'8"
7'4"

Every step adds value

Organize code

Every step adds value

Organize code

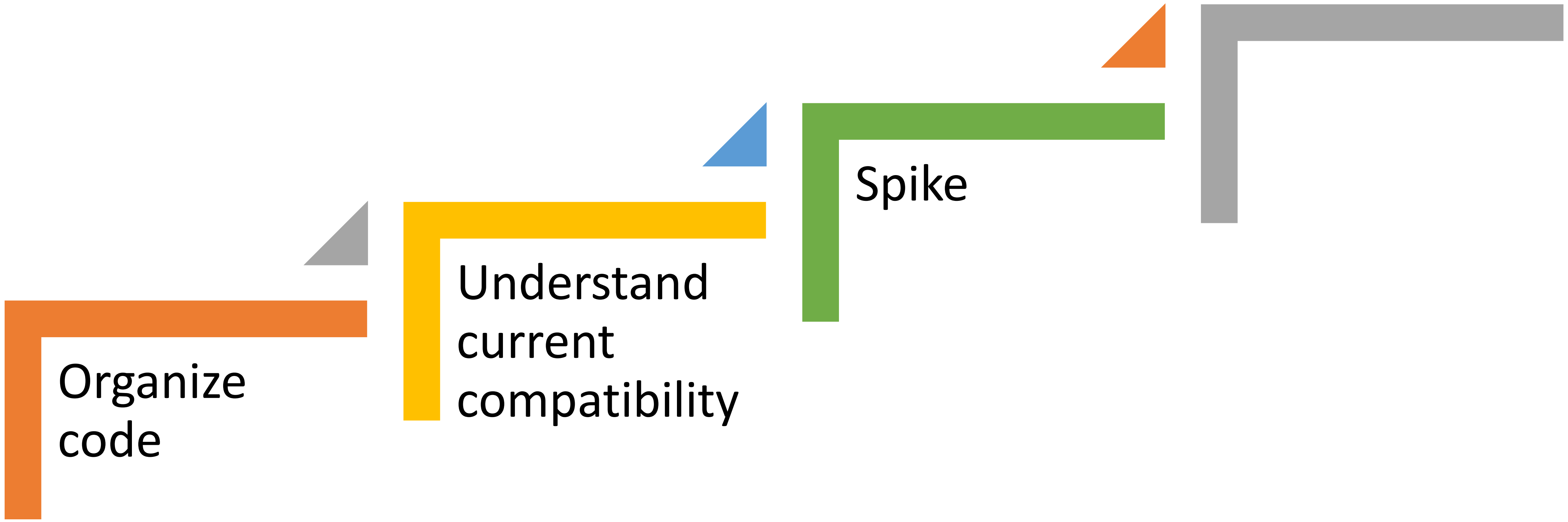
Understand current compatibility

Figuring Out Compatibility

Demo

- apisof.net
- .NET PortabilityAnalyzer
  - VS extension and EXE for command – in Gallery
- API Analyzer
  - Analyzer downloaded via NuGet – Microsoft.DotNet.Analyzers.Compatibility

Every step adds value



Every step adds value

Organize code

Understand current compatibility

Spike

Standard when/where needed

Standard When/Where Needed

- Standard is awesome!
  - Allows you to write for cross platform
  - Maintains a broad and effective API surface (2.0)
  - Has a broad reach (1.n)
  - Defines a common future
- Your projects can have multiple targets
  - Compile for .NET Standard for most platforms
  - Also, compile for .NET Full Framework if that makes sense

Every step adds value

Organize code

Understand current compatibility

Spike

Standard when/where needed