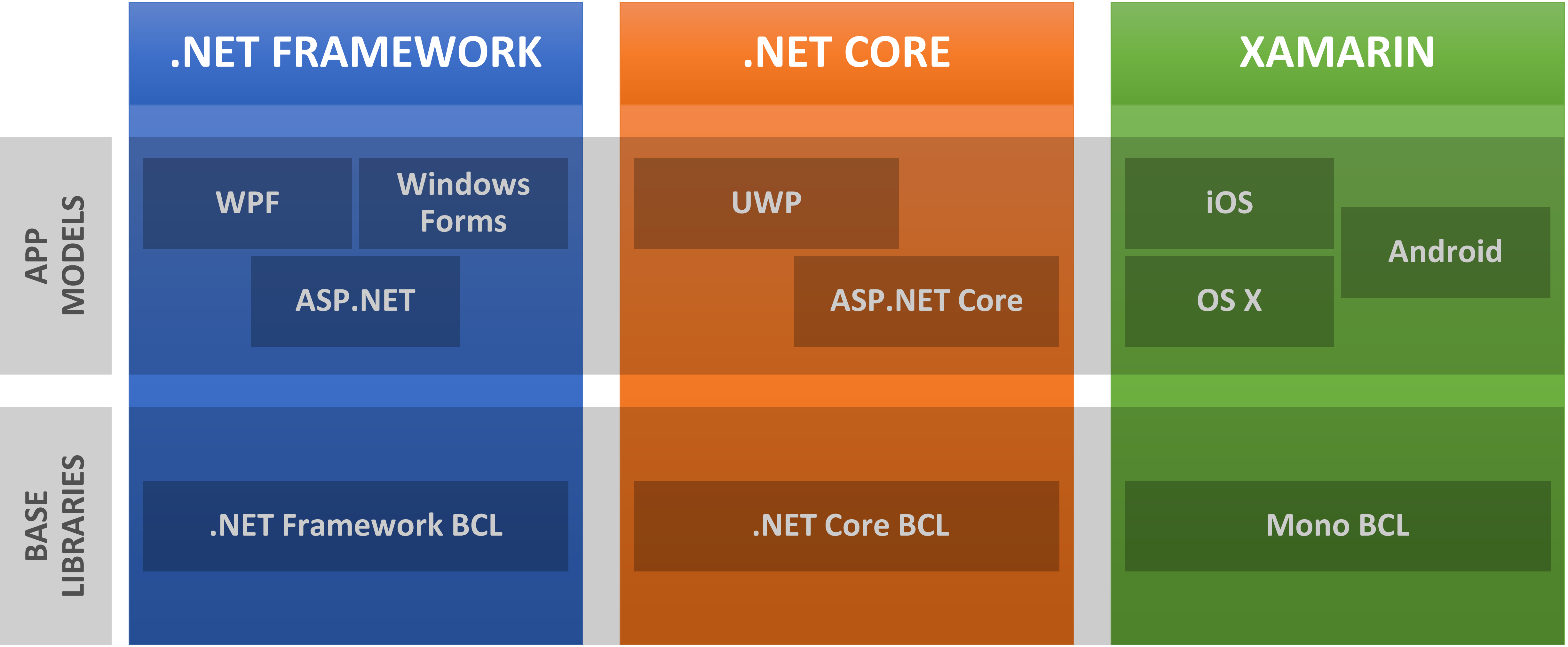
Microsoft

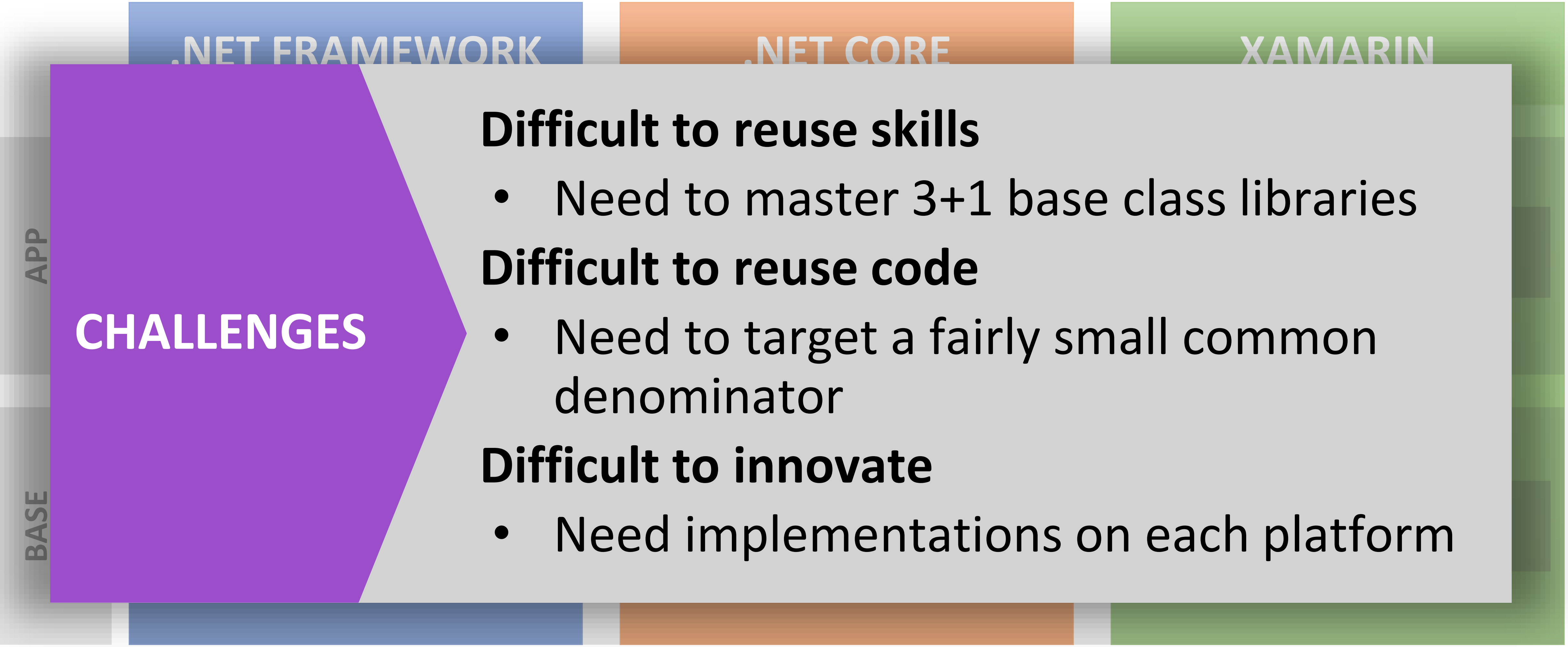
.NET Standard

.NET today—reusing code



	.NET FRAMEWORK	.NET CORE	XAMARIN
APP MODELS	WPF Windows Forms ASP.NET	UWP ASP.NET Core	iOS OS X Android
BASE LIBRARIES	.NET Framework BCL	.NET Core BCL	Mono BCL

.NET today—reusing code



CHALLENGES

Difficult to reuse skills

- Need to master 3+1 base class libraries

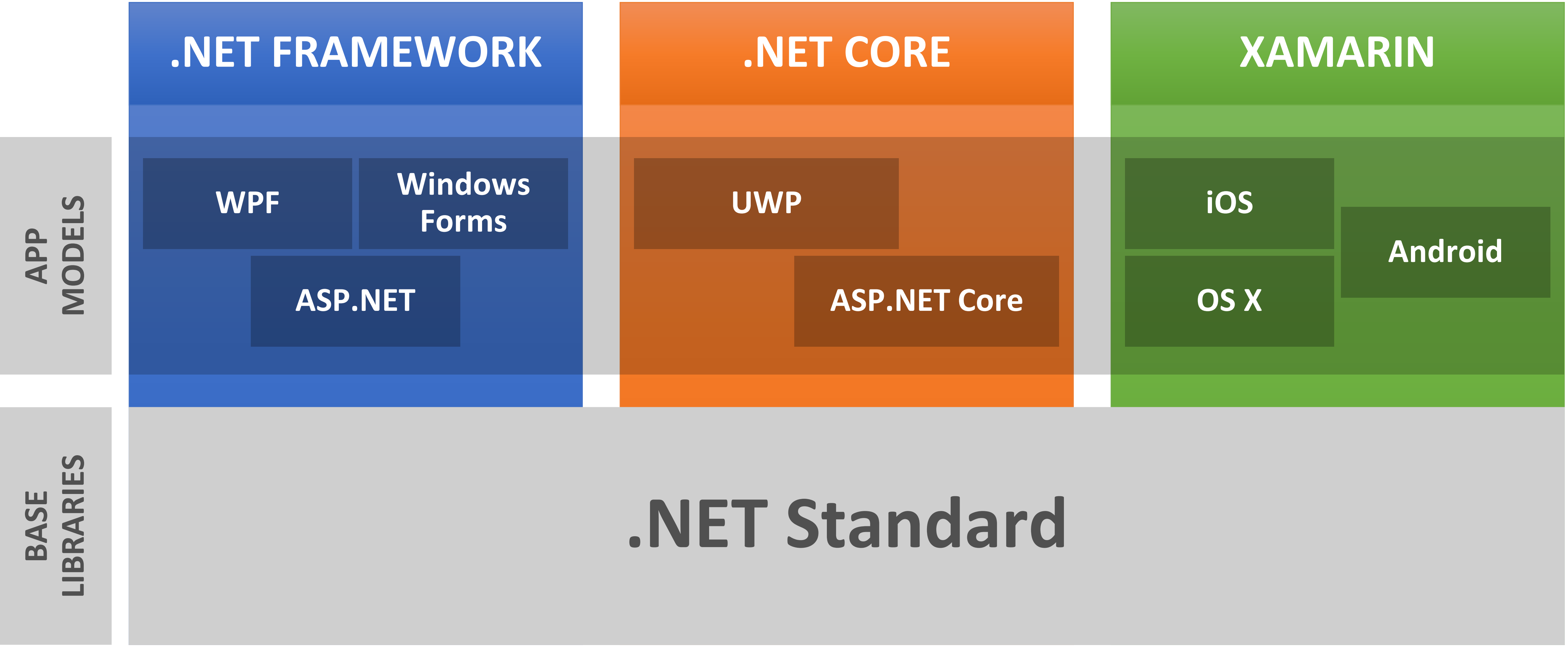
Difficult to reuse code

- Need to target a fairly small common denominator

Difficult to innovate

- Need implementations on each platform

.NET tomorrow



APP MODELS

BASE LIBRARIES

.NET FRAMEWORK

WPF

Windows Forms

ASP.NET

.NET CORE

UWP

ASP.NET Core

XAMARIN

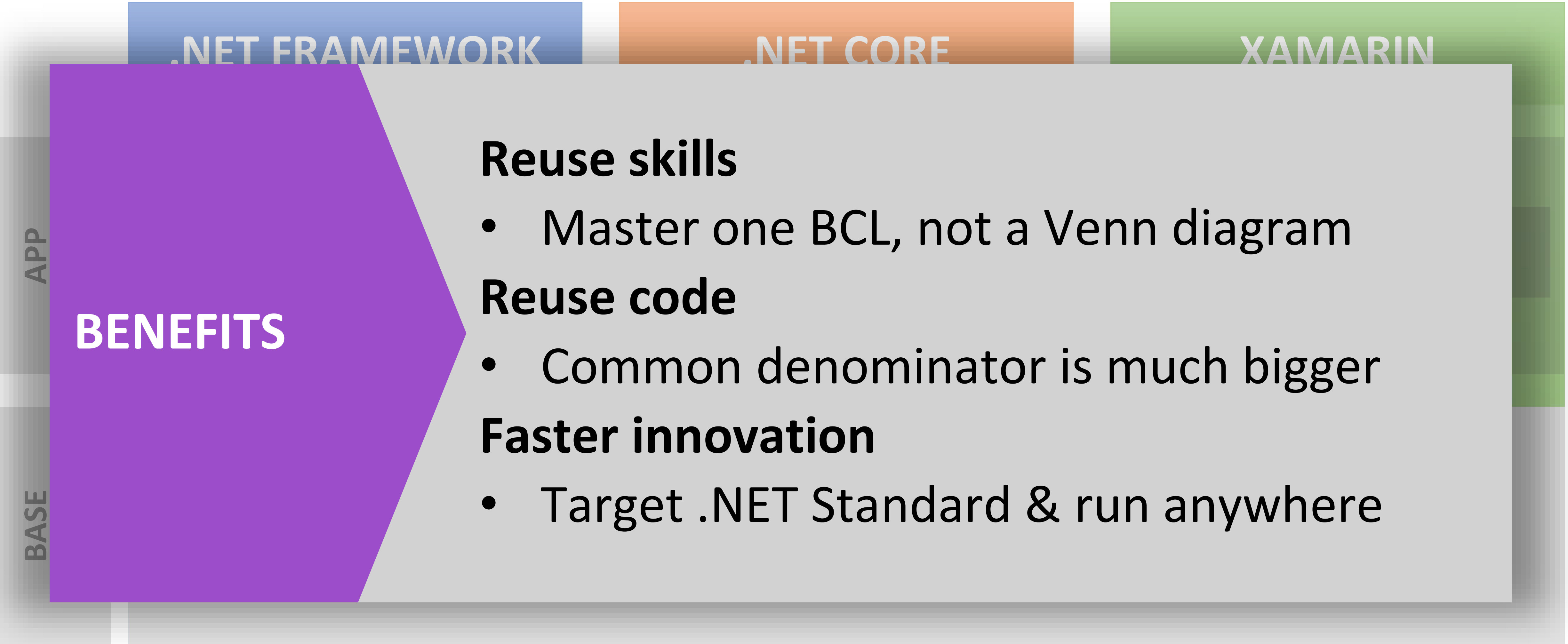
iOS

OS X

Android

.NET Standard

.NET tomorrow



.NET FRAMEWORK

.NET CORE

XAMARIN

APP

BASE

BENEFITS

Reuse skills

- Master one BCL, not a Venn diagram

Reuse code

- Common denominator is much bigger

Faster innovation

- Target .NET Standard & run anywhere

What is .NET Standard?

- .NET Standard **is a specification**
- A set of APIs that **all .NET platforms have to implement**

.NET Standard	**~**	**HTML specification**
.NET Framework **.NET Core** **Xamarin**	**~**	**Browsers**

.NET Standard 2.0

Has much bigger API surface

- Extended to cover intersection between .NET Framework and Xamarin
- 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

- Compat shim allows referencing existing .NET Framework code – without recompilation
- Limited to libs that use APIs that are available for .NET Standard

~70%

of NuGet packages are API compatible

What version should you target?

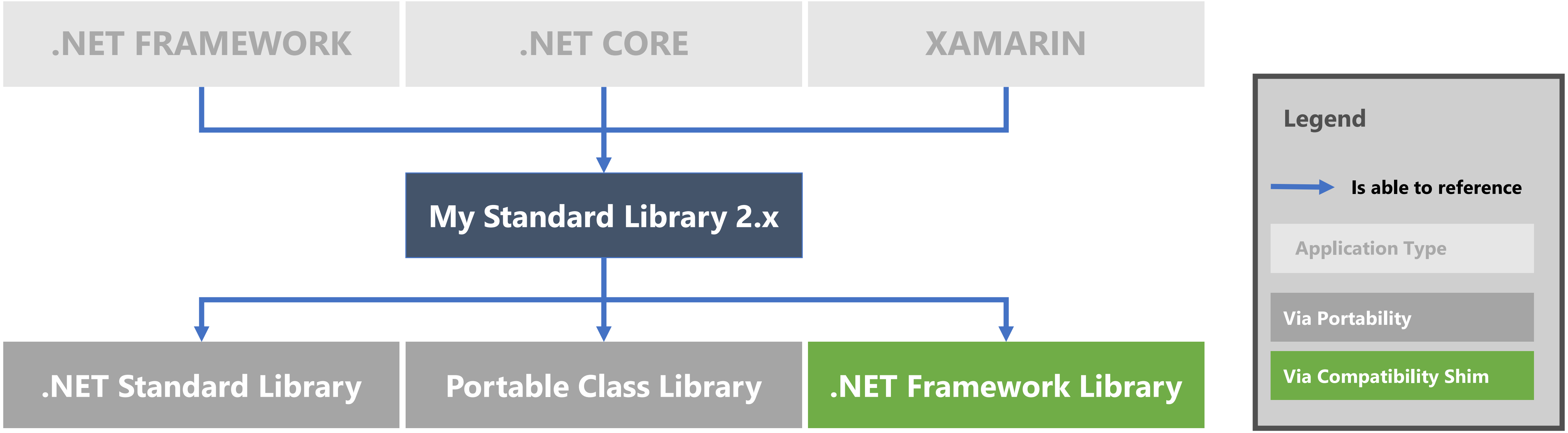
- The **higher the version**, the **more APIs** you ha
- The **lower the version**, the mo

Target the lowest version you can get away with!

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

What can you reference from .NET Standard?



.NET FRAMEWORK

.NET CORE

XAMARIN

My Standard Library 2.x

.NET Standard Library

Portable Class Library

.NET Framework Library

Legend

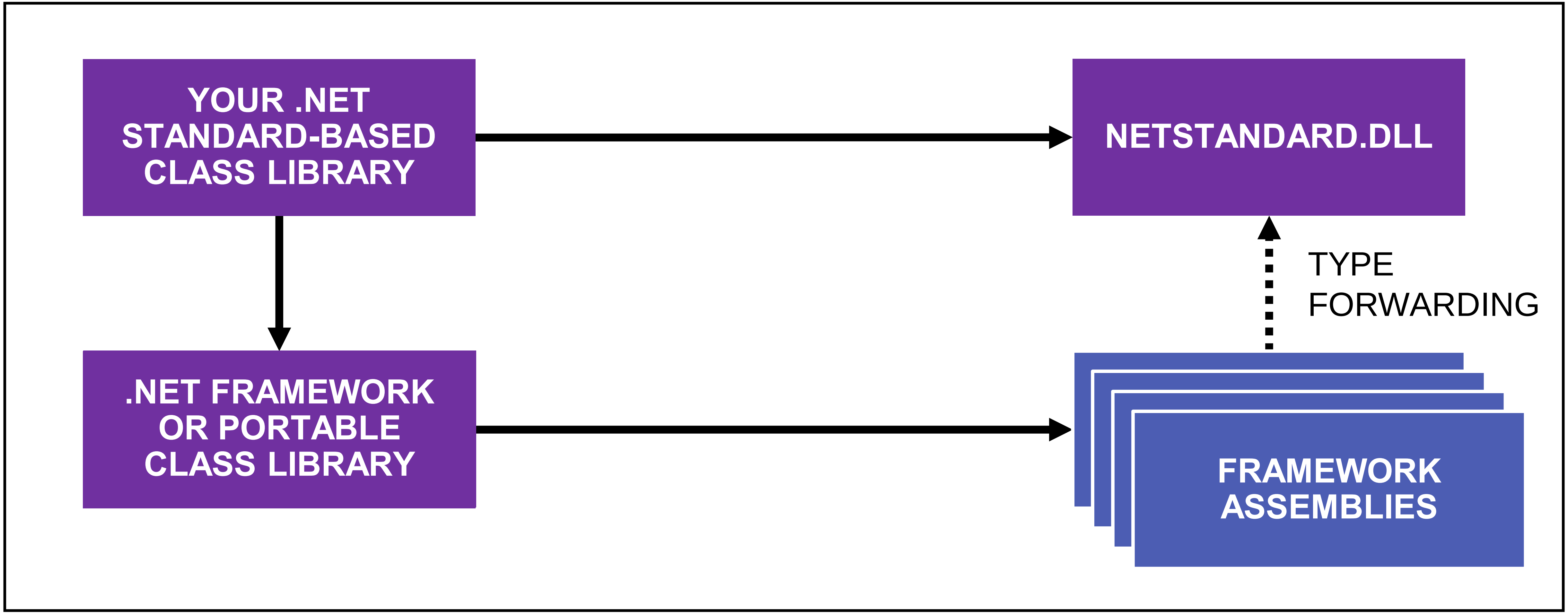
→ Is able to reference

Application Type

Via Portability

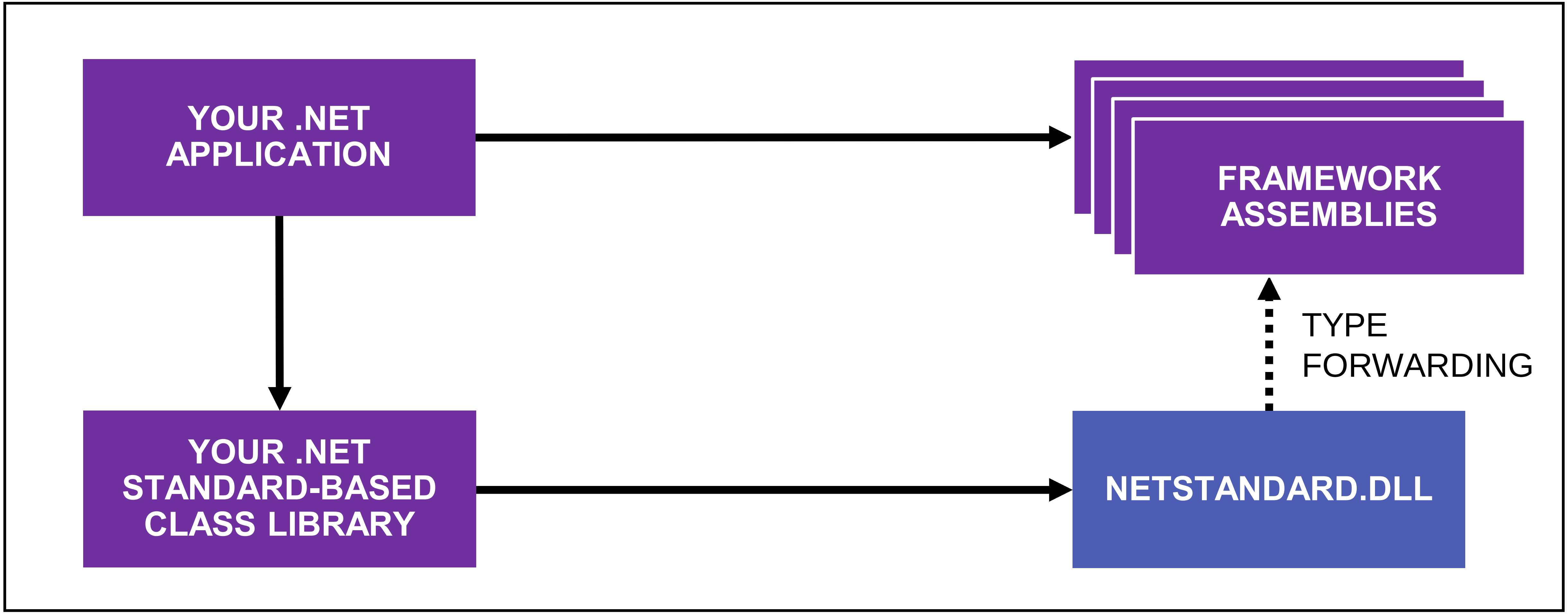
Via Compatibility Shim

.NET Standard under the hood


YOUR .NET STANDARD-BASED CLASS LIBRARY
NETSTANDARD.DLL
TYPE FORWARDING
.NET FRAMEWORK OR PORTABLE CLASS LIBRARY
FRAMEWORK ASSEMBLIES

This happens when you build a .NET Standard-based Library

.NET Standard under the hood

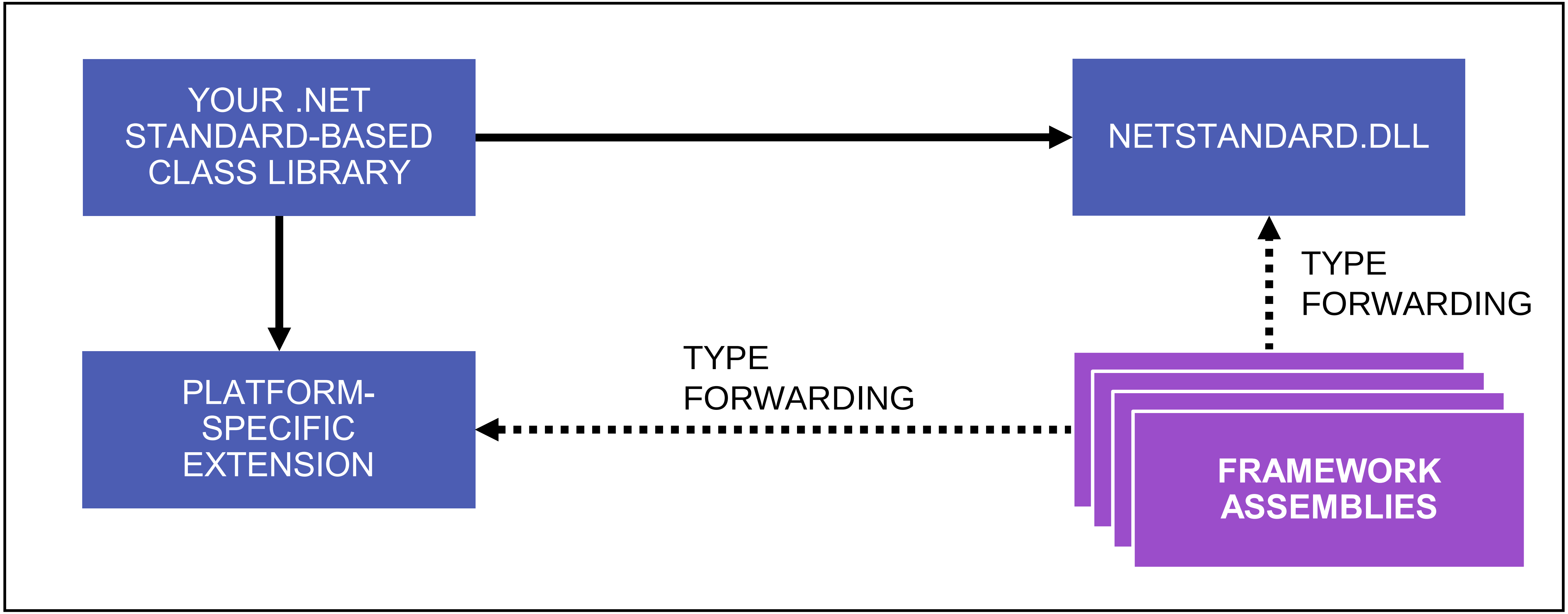


This happens when you load .NET Standard-based library

Platform specific APIs & .NET Standard

- .NET Standard (mostly) only **contains APIs that will work everywhere**
  - We generally avoid adding large chunks of APIs that don’t work everywhere
  - A small set of APIs will throw PlatformNotSupportedException
- Platform specific APIs sit **on top of .NET Standard** & you **can add references** to them
  - Examples: Registry, Reflection Emit, Access Control, Windows Identity
  - You’ll become less portable

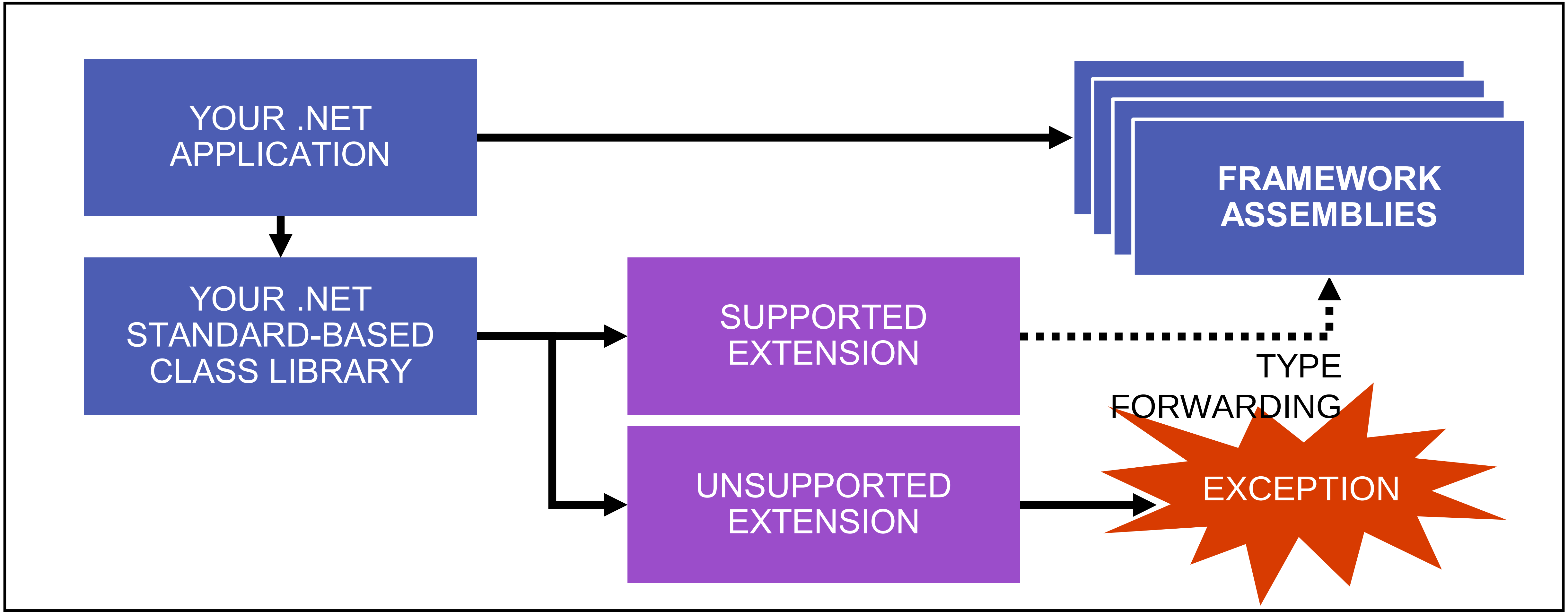
Platform specific APIs & .NET Standard



YOUR .NET STANDARD-BASED CLASS LIBRARY
NETSTANDARD.DLL
TYPE FORWARDING
PLATFORM-SPECIFIC EXTENSION
TYPE FORWARDING
FRAMEWORK ASSEMBLIES

This happens when you build a .NET Standard-based library with platform-specific extensions

Platform specific APIs & .NET Standard



This happens when you load .NET Standard-based library with platform-specific extensions

What about the breaking change?

- .NET Framework 4.6.1 will have the broadest a   ship
  - Doesn't support all the APIs in 1.6 (
  - Does supports most of
- We con   :

No breaking change between .NET Standard 1.x and 2.0!

.NET St			1.2	1.3	1.4	1.5	1.6	2.0
.NET Fra ork	→	4.5	4.5.1	4.6	4.6.1	4.6.2	vNext	4.6.1
.NET Framework	→	4.5	4.5.1	4.6	→	→	→	4.6.1

What’s new in .NET Standard 2.0?

Many more APIs!

- .NET standard 2.0 more than doubles the number of APIs!

Version	#APIs	Growth %
1.x	13,501	+1%
2.0	32,638	+142%

Compat with .NET Framework libs!

- Most libraries are still targeting .NET Framework

Target	Usage on NuGet
.NET Framework	46,894
.NET Standard	1,886
PCL	4,501

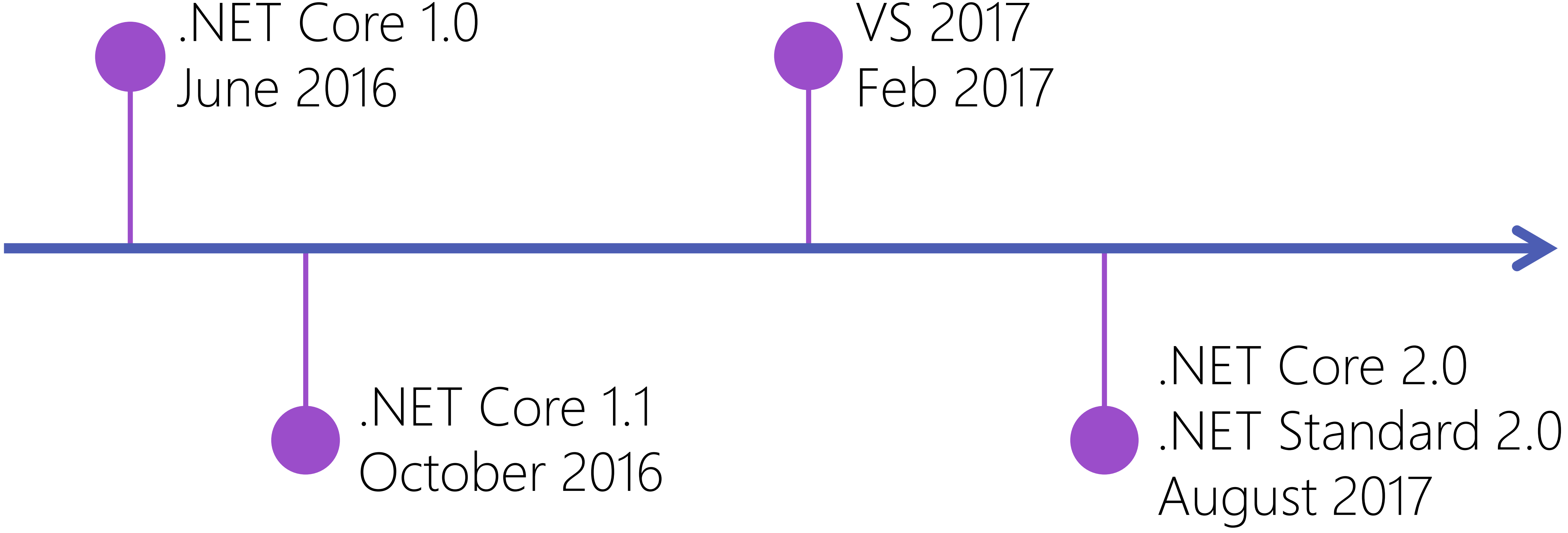
- A compat shim makes them usable on other platforms, with caveats

.NET Core and .NET Standard

- **.NET Core is an implementation** of the .NET Standard
- They are **fully separated**, e.g. different GitHub repositories

- **.NET Standard updates are coordinated** across all .NET implementers
  - There is a .NET Standard review board
- **.NET Core can be updated independently**
  - Used by us to experiment and accelerate innovation

.NET Core & .NET Standard Releases



.NET Core 1.0
June 2016

.NET Core 1.1
October 2016

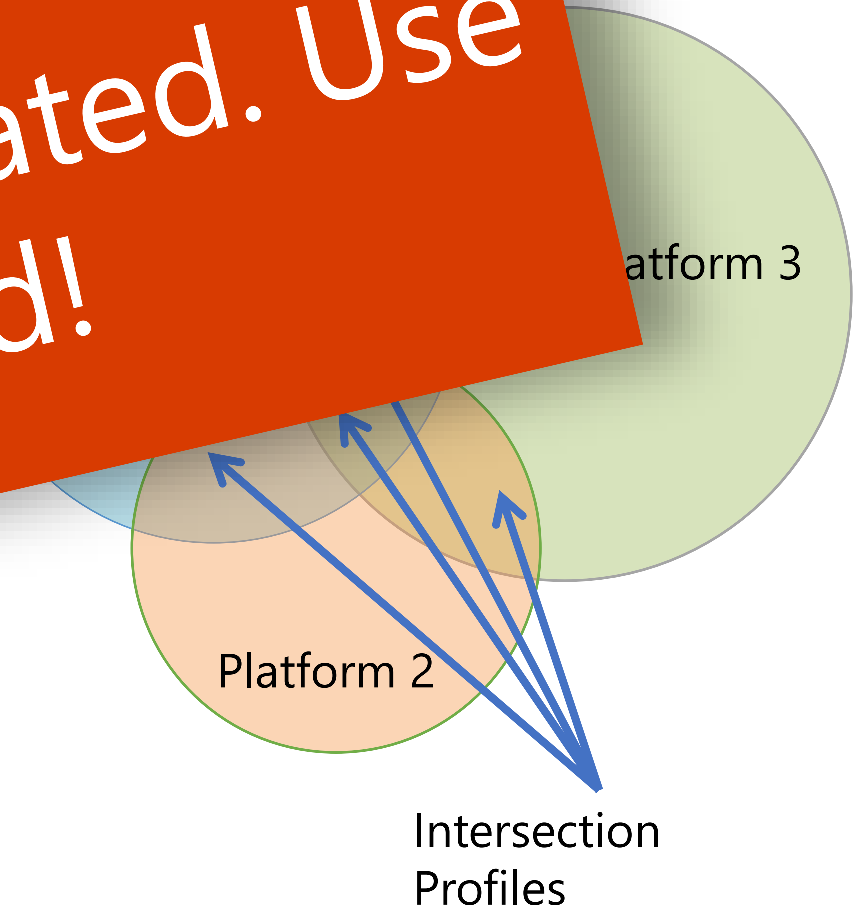
VS 2017
Feb 2017

.NET Core 2.0
.NET Standard 2.0
August 2017

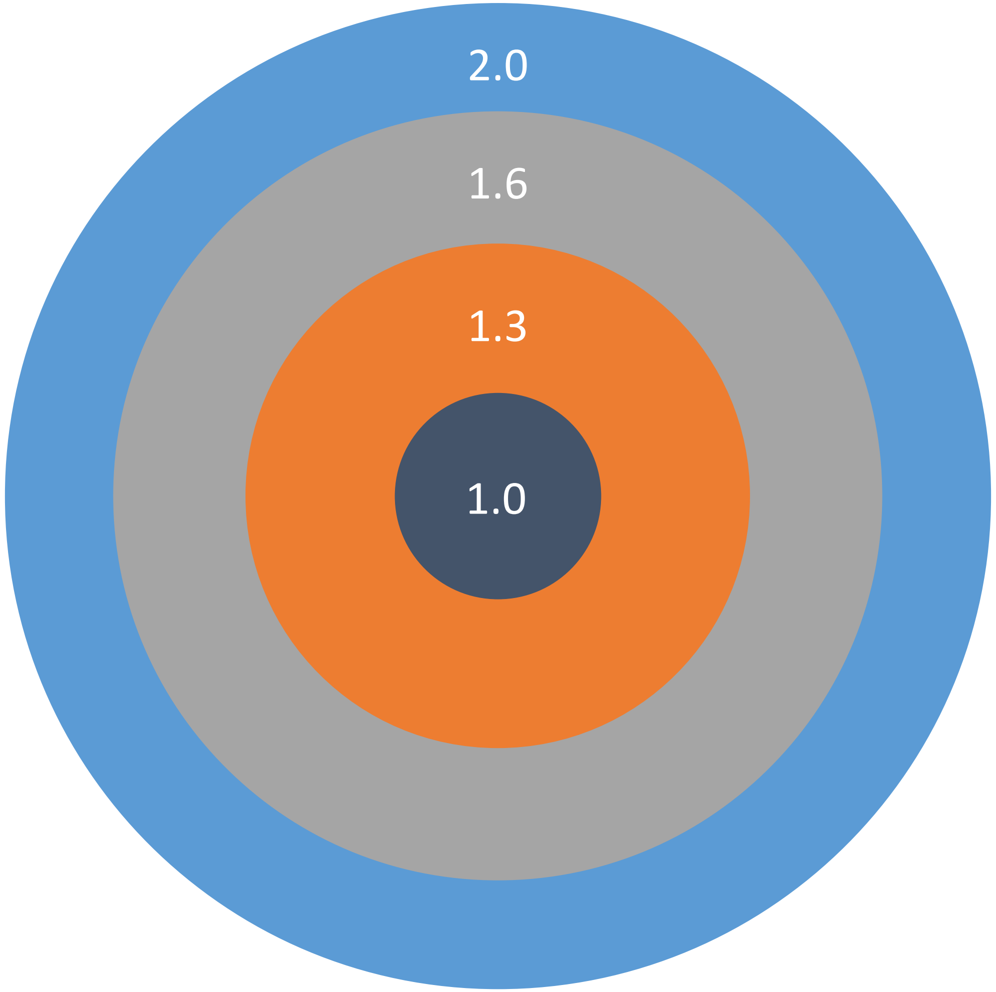
Difference to Portable Class Libraries (PCL)

- PCLs were an after thought, i.e. each platform ch APIs to includes
  - No systematic appro
  - Comp
- E pl
  - platforms
  - stand compatibility relationships

PCLs are now deprecated. Use .NET Standard!



Versioning 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