



# .NET, C#, Visual Studio

## Ein Rundumblick

Christian Schwendtner

SNEK 6, 22.4.2018



LOOK

# Agenda

.NET Core

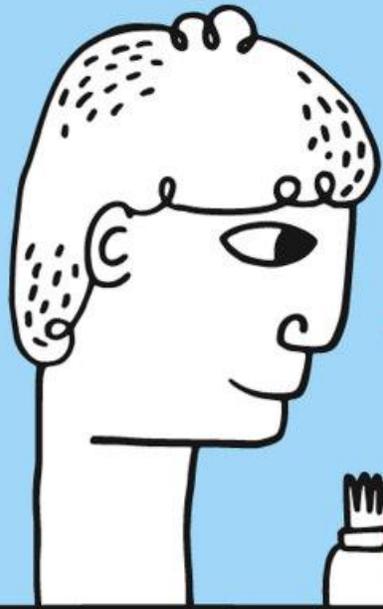
.NET Standard

.NET Core / .NET (Full) Framework

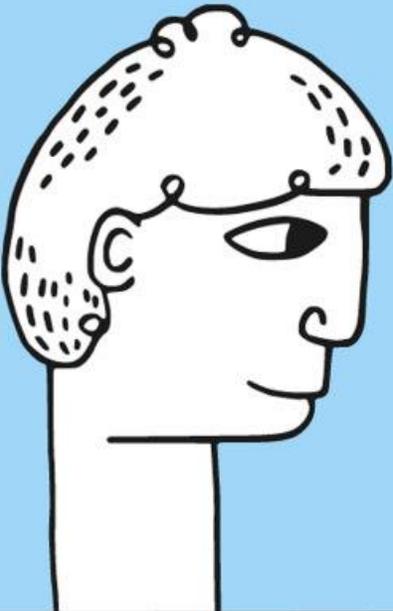
C# 7

Visual Studio 2017

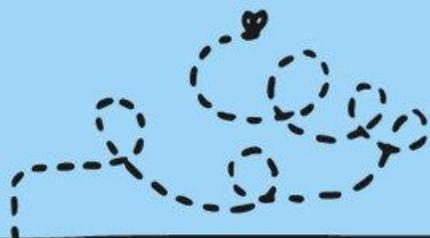




WHO WANTS CHANGE?



WHO WANTS TO CHANGE?





# Microsoft verändert sich (noch immer)

## Selected Product and Service Revenue Constant Currency Reconciliation

|  | Three Months Ended           |
|--|------------------------------|
|  | Percentage Change Y/Y (GAAP) |
| Office commercial products and cloud services          | 10%                          |
| Office 365 commercial                                  | 41%                          |
| Office consumer products and cloud services            | 12%                          |
| Dynamics products and cloud services                   | 10%                          |
| Dynamics 365   | 67%                          |
| Server products and cloud services                     | 18%                          |
| Azure  | 98%                          |
| Enterprise Services                                    | 5%                           |
| Windows OEM  | 4%                           |
| Windows commercial products and cloud services         | (4)%                         |
| Search advertising excluding traffic acquisition costs | 15%                          |
| Surface  | 1%                           |
| Gaming   | 8%                           |

# Begriffe

.NET

.NET Platform

.NET Implementation

.NET Framework

.NET Runtime

.NET Framework  $\neq$  .NET Core  $\neq$  Xamarin

.NET Framework

= ".NET Full Framework"

= .NET 1.x – 4.x

= "was wir bisher kannten"

**.NET FRAMEWORK**

**.NET CORE**

**XAMARIN**

<https://docs.microsoft.com/en-us/dotnet/standard/glossary>



# .NET Core

Plattformunabhängig

Windows, macOS, Linux

Leichtgewichtig

Schnell

Open Source

# .NET & Open Source

○ 2001- ECMA-335

○ 2002- .NET 1.0 for Windows released, Mono project begins

○ 2008- ASP.NET MVC (web platform) Open Source

○ .NET foundation  
April 2014- .NET Compiler Platform ("Roslyn") Open Source  
.NET Foundation founded

○ Nov 2014- .NET Core Cross-plat, Open Source



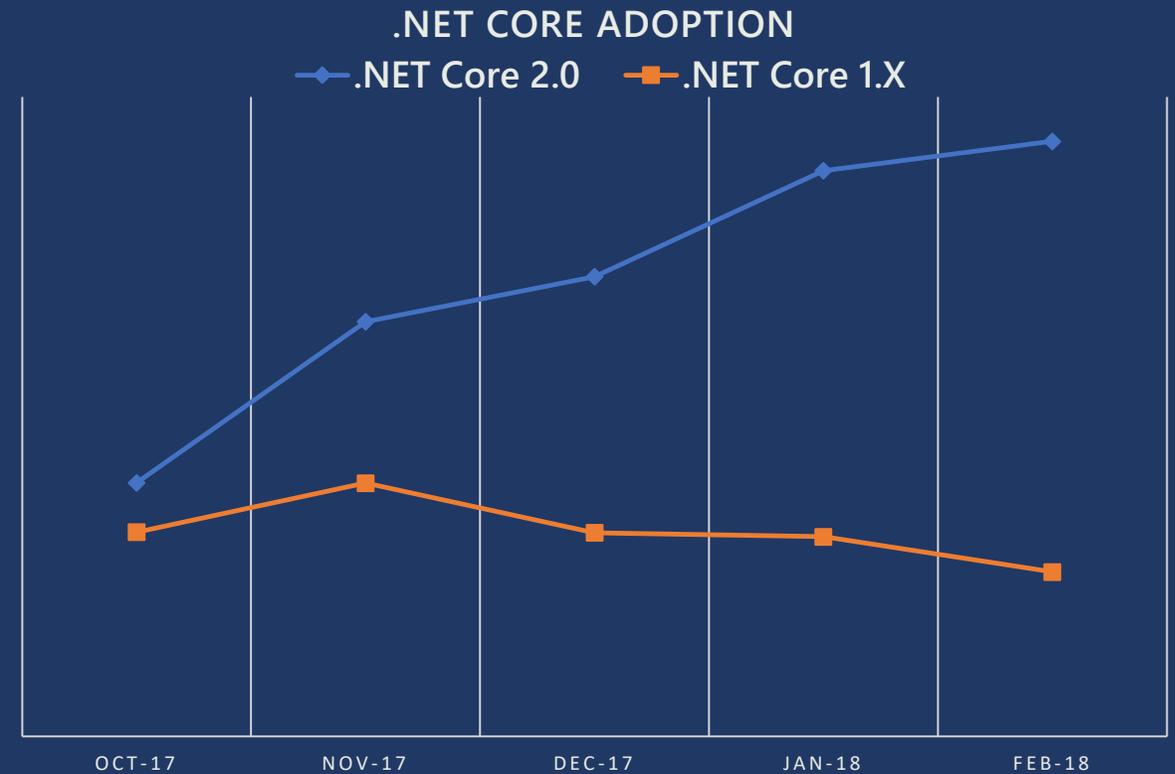
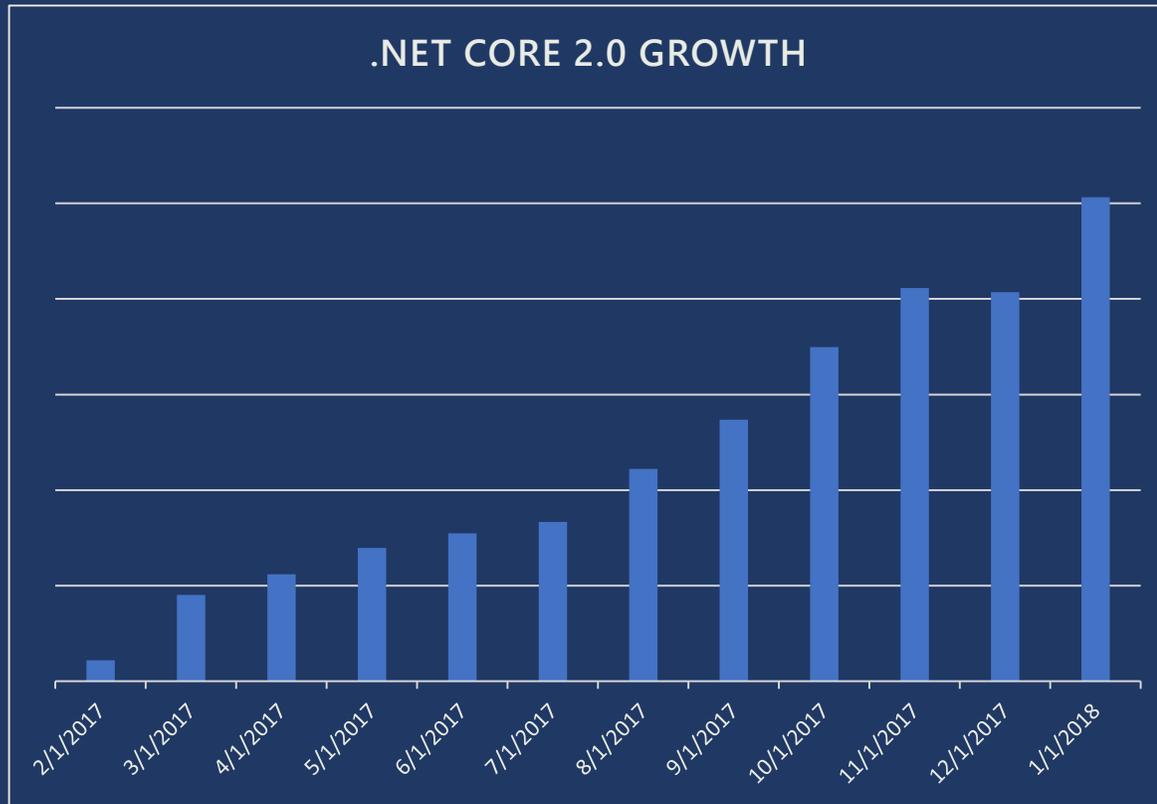
○ 2016- Mono project joins .NET Foundation



○ Aug 2017- .NET Core 2

# .NET Core

## Over Half Million Active\* .NET Core 2.0 Developers!



\* Active = unique monthly developers with 2+ days of development per month.

# .NET Core

CoreCLR – .NET Core Runtime

CoreFX – .NET Core Framework Libraries

Roslyn

**dotnet** – .NET Core CLI – SDK Tools

**dotnet** – Application Host

# Anwendungstypen

Console Application

ASP.NET Core Application

UWP

Xamarin.Forms

# Unterstützte Betriebssysteme

## Windows

| OS                | Version       | Architectures |
|-------------------|---------------|---------------|
| Windows Client    | 7 SP1+, 8.1   | x64, x86      |
| Windows 10 Client | Version 1607+ | x64, x86      |
| Windows Server    | 2008 R2 SP1+  | x64, x86      |

## macOS

| OS       | Version | Architectures |
|----------|---------|---------------|
| Mac OS X | 10.12+  | x64           |

## Linux

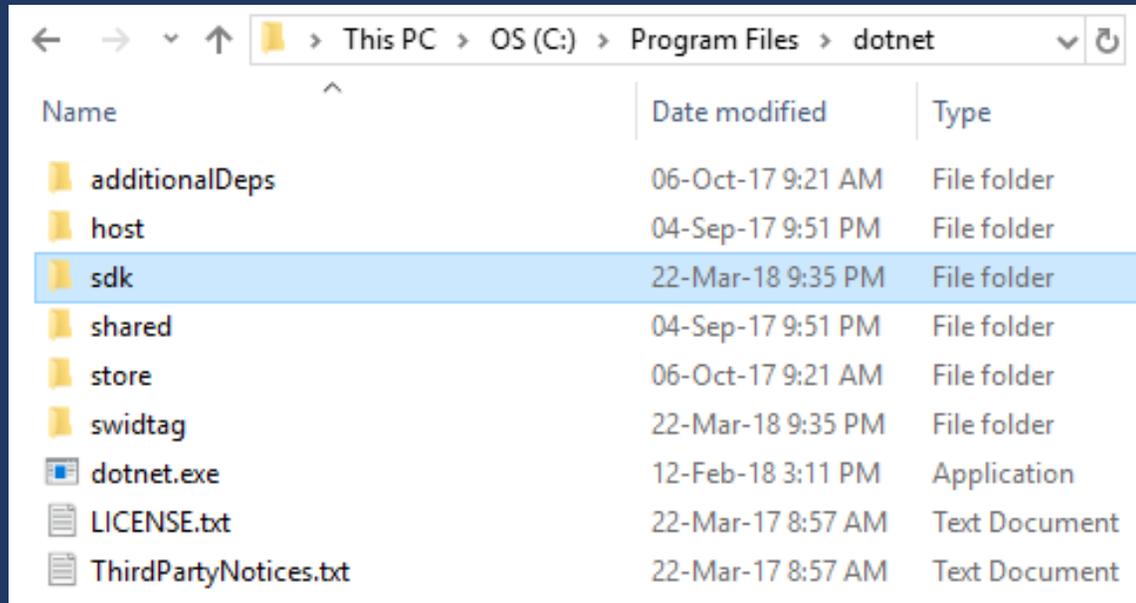
| OS                           | Version             | Architectures |
|------------------------------|---------------------|---------------|
| Red Hat Enterprise Linux     | 7                   | x64           |
| CentOS                       |                     |               |
| Oracle Linux                 |                     |               |
| Fedora                       | 26, 27              | x64           |
| Debian                       | 9, 8.7+             | x64           |
| Ubuntu                       | 17.10, 16.04, 14.04 | x64           |
| Linux Mint                   | 18, 17              |               |
| openSUSE                     | 42.2+               | x64           |
| SUSE Enterprise Linux (SLES) | 12                  | x64           |

<https://github.com/dotnet/core/blob/master/release-notes/2.0/2.0-supported-os.md>

*'Support' means that .NET Core is built and tested on the OS and Microsoft Developer Support may be contacted for assistance with .NET Core on the environment.*

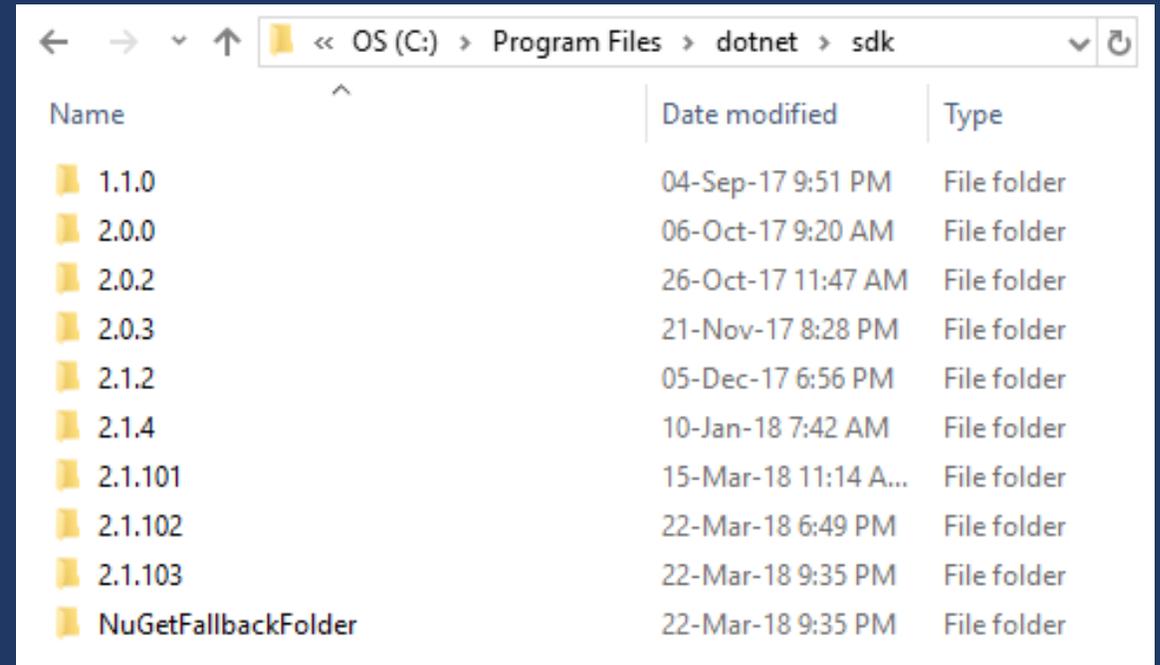
<https://github.com/dotnet/core/blob/master/os-lifecycle-policy.md>

# Side-by-side



← → ▾ ↑  > This PC > OS (C:) > Program Files > dotnet ▾ ↻

| Name  | Date modified     | Type          |
|---|-------------------|---------------|
|  additionalDeps        | 06-Oct-17 9:21 AM | File folder   |
|  host                  | 04-Sep-17 9:51 PM | File folder   |
|  sdk                   | 22-Mar-18 9:35 PM | File folder   |
|  shared                | 04-Sep-17 9:51 PM | File folder   |
|  store                 | 06-Oct-17 9:21 AM | File folder   |
|  swidtag               | 22-Mar-18 9:35 PM | File folder   |
|  dotnet.exe            | 12-Feb-18 3:11 PM | Application   |
|  LICENSE.txt           | 22-Mar-17 8:57 AM | Text Document |
|  ThirdPartyNotices.txt | 22-Mar-17 8:57 AM | Text Document |



← → ▾ ↑  << OS (C:) > Program Files > dotnet > sdk ▾ ↻

| Name  | Date modified        | Type        |
|---|----------------------|-------------|
|  1.1.0                 | 04-Sep-17 9:51 PM    | File folder |
|  2.0.0                 | 06-Oct-17 9:20 AM    | File folder |
|  2.0.2                 | 26-Oct-17 11:47 AM   | File folder |
|  2.0.3                 | 21-Nov-17 8:28 PM    | File folder |
|  2.1.2               | 05-Dec-17 6:56 PM    | File folder |
|  2.1.4               | 10-Jan-18 7:42 AM    | File folder |
|  2.1.101             | 15-Mar-18 11:14 A... | File folder |
|  2.1.102             | 22-Mar-18 6:49 PM    | File folder |
|  2.1.103             | 22-Mar-18 9:35 PM    | File folder |
|  NuGetFallbackFolder | 22-Mar-18 9:35 PM    | File folder |

# Frameworks / Packages / Metapackages

## Frameworks

.NET Framework, Version=4.6

.NET Core, Version=2.0

.NET Standard, Version=2.0

## Packages – NuGet Packages

*System.Runtime*

*System.Collections*

*System.Net.Http*

...

## Metapackage(s) – Set von Packages

*Microsoft.NETCore.App*

*Microsoft.AspNetCore.All*

...

| Target Framework      | Latest Version | Target Framework Moniker (TFM) |
|-----------------------|----------------|--------------------------------|
| .NET Standard         | 2.0            | netstandard2.0                 |
| .NET Core Application | 2.0            | netcoreapp2.0                  |
| .NET Framework        | 4.7.1          | net471                         |



“Alles ist neu“ (verglichen mit 2016 ;)

~~DNVM~~

~~DNU~~

**dotnet** / .NET Core CLI

~~DNX~~

~~project.json~~

csproj

dotnet

“Driver”

CLI

```
dotnet new console
```

Application Host (FDD)

```
dotnet MyApp.dll
```

# dotnet CLI

## dotnet

|                |   |
|----------------|---|
| <b>new</b>     | Neues Projekt anlegen                     |
| <b>restore</b> | Abhängigkeiten (NuGet) (wieder)herstellen |
| <b>build</b>   | Anwendung übersetzen / erstellen          |
| <b>run</b>     | Anwendung ausführen (mit Quellcode)       |
| <b>test</b>    | Tests ausführen                           |
| <b>pack</b>    | NuGet-Paket erstellen                     |
| <b>publish</b> | Anwendung (+Abhängigkeiten) in Ordner     |
| <b>migrate</b> | Project.json -> msbuild (csproj) Projekt  |

# .csproj – .NET Core Class Library

```
<Project Sdk="Microsoft.NET.Sdk">
```

```
  <PropertyGroup>
```

```
    <TargetFramework>netcoreapp2.0</TargetFramework>
```

```
  </PropertyGroup>
```

```
</Project>
```

.csproj – EXE

```
<Project Sdk="Microsoft.NET.Sdk">
```

```
  <PropertyGroup>
```

```
    <OutputType>Exe</OutputType>
```

```
    <TargetFramework>netcoreapp2.0</TargetFramework>
```

```
  </PropertyGroup>
```

```
</Project>
```

# .csproj – .NET Standard Class Library

```
<Project Sdk="Microsoft.NET.Sdk">
```

```
  <PropertyGroup>
```

```
    <TargetFramework>netstandard2.0</TargetFramework>
```

```
  </PropertyGroup>
```

```
</Project>
```

## .csproj – Multi-Targeting

```
<Project Sdk="Microsoft.NET.Sdk">  
  <PropertyGroup>  
    <TargetFrameworks>netcoreapp2.0;net47</TargetFrameworks>  
  </PropertyGroup>  
  
</Project>
```

# .csproj – Multi-Targeting

```
<Project Sdk="Microsoft.NET.Sdk">  
  <PropertyGroup>  
    <TargetFrameworks>netstandard1.4;net40;net45</TargetFrameworks>  
  </PropertyGroup>  
  
  <ItemGroup Condition=" '$(TargetFramework)' == 'net40' ">  
    <Reference Include="System.Net" />  
  </ItemGroup>  
  
  <ItemGroup Condition=" '$(TargetFramework)' == 'net45' ">  
    <Reference Include="System.Net.Http" />  
    <Reference Include="System.Threading.Tasks" />  
  </ItemGroup>  
  
</Project>
```

# Target Framework Monikers (TFMs)

| Target Framework | TFM            |
|------------------|----------------|
| .NET Standard    | netstandard1.0 |
|                  | netstandard1.1 |
|                  | netstandard1.2 |
|                  | netstandard1.3 |
|                  | netstandard1.4 |
|                  | netstandard1.5 |
|                  | netstandard1.6 |
|                  | netstandard2.0 |
| .NET Core        | netcoreapp1.0  |
|                  | netcoreapp1.1  |
|                  | netcoreapp2.0  |
| .NET Framework   | net11          |
|                  | net20          |
|                  | net35          |
|                  | net40          |
|                  | net403         |
|                  | net45          |
|                  | net451         |
|                  | net452         |
|                  | net46          |
|                  | net461         |
|                  | net462         |
|                  | net47          |
|                  | net471         |

<https://docs.microsoft.com/en-us/dotnet/standard/frameworks>

dotnet run / build

dotnet run / build

--configuration *Debug|Release*

--framework *framework*

# Deployment Varianten

Framework-Dependent Deployment (FDD)

Systemweite .NET Core Version muss vorhanden sein

Self-Contained Deployment (SCD)

Deployment "enthält" .NET Core Runtime

Framework Dependent Deployment  
(FDD)

.NET Runtime

Self-contained Deployment  
(SCD)

.NET Runtime

# dotnet publish

dotnet publish

--configuration *Debug|Release*

--framework *framework*

--self-contained

--runtime *RID*

RID – Runtime IDentifier

win-x86

win-x64

linux-x64

osx-x64

win10-x64

debian.8-x64

...

# Framework-dependent Deployment

```
dotnet publish -c Release
```

“Framework-dependent app deployment”

Zentrale .NET Core Runtime

Apps unabhängig von Betriebssystem (und Rechner-Architektur)

Gut geeignet für “kontrollierbare” Umgebungen -> Platzbedarf, Wartung

Aufruf mithilfe des .NET Core Hosts: `dotnet some/path/myApp.dll`

# Self-contained Deployment

```
dotnet publish --self-contained -c Release -r win-x64
```

“self contained app deployment”

Jede App hat eigene private .NET Core Runtime -> Platzbedarf

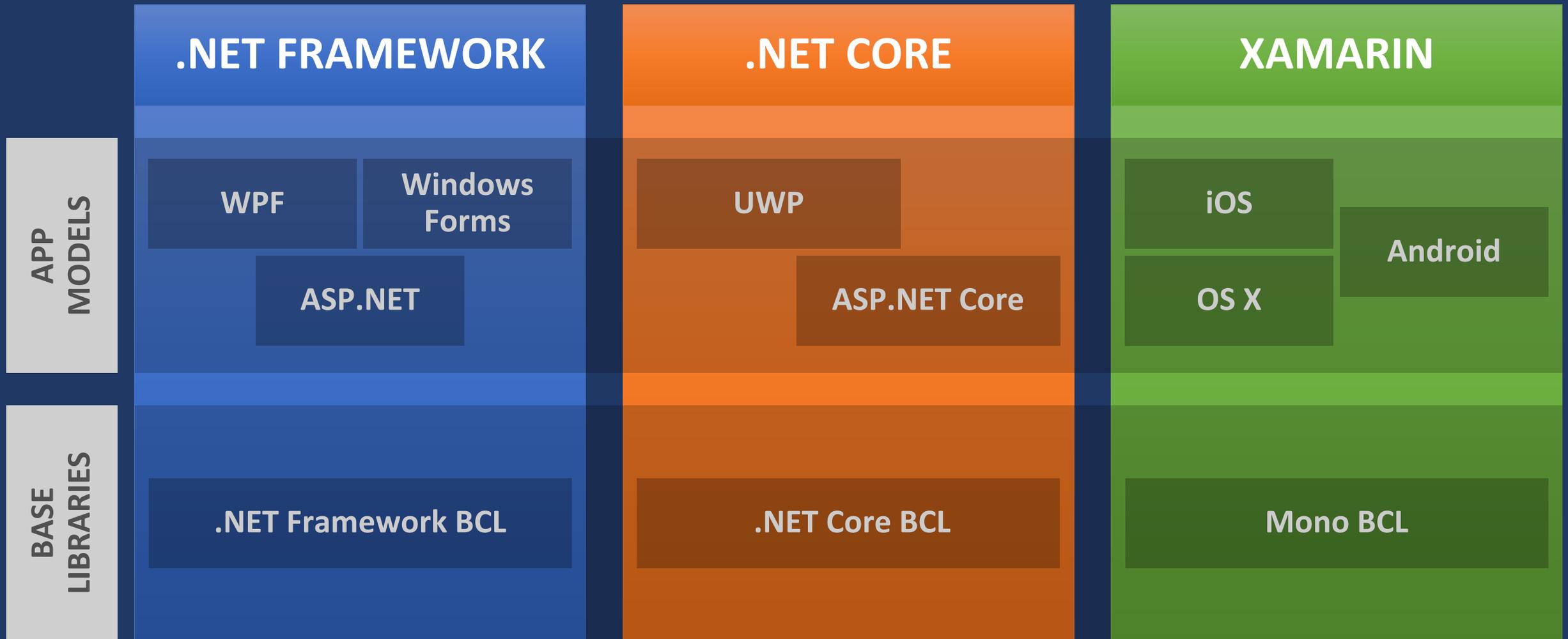
App abhängig vom Betriebssystem (und Rechner-Architektur)

Gut geeignet für “nicht kontrollierbare” Umgebungen

Aufruf als (spezifisches) Executable **myApp.exe**



# .NET – “bisher”



# ECMA 335

ECMA 335 continues to establish uniformity for .NET implementation behavior, but **there is no similar spec for the .NET Base Class Libraries (BCL)** for .NET library implementations.

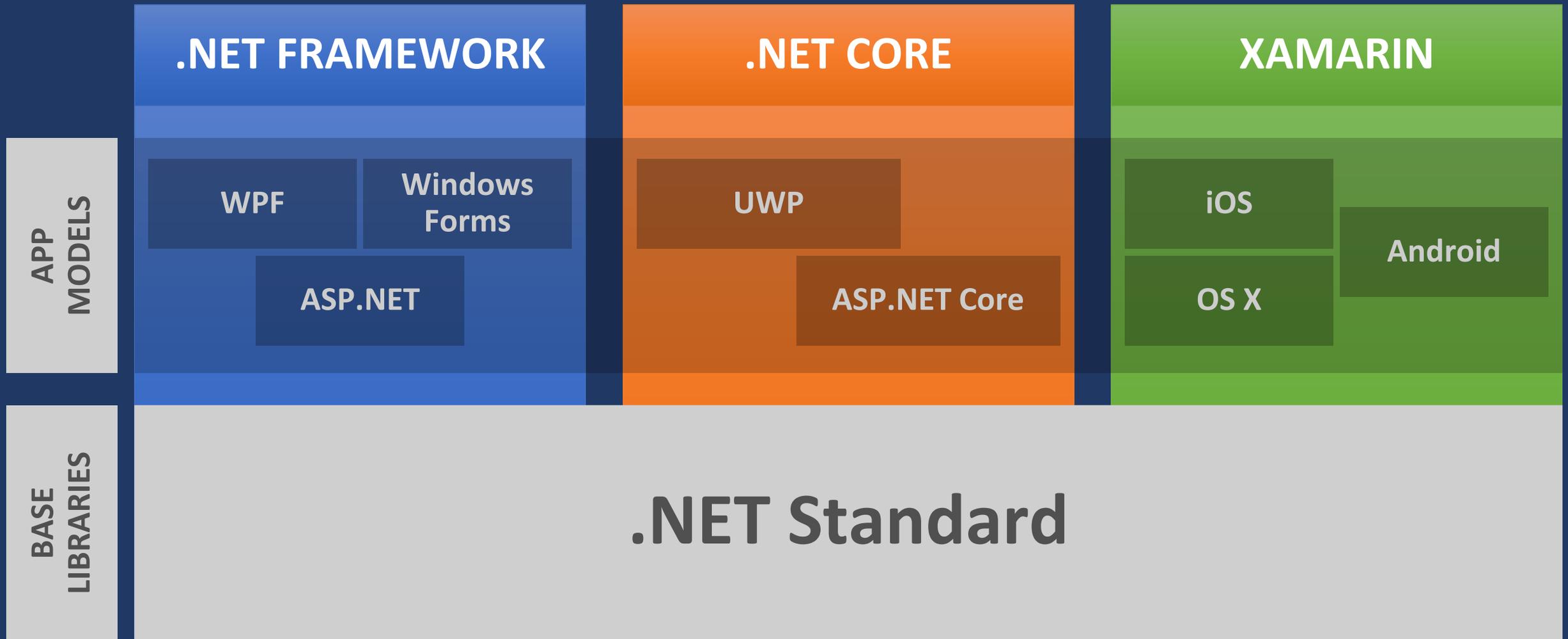
# .NET – “bisher”

Verschiedene BCLs

Code-Wiederverwendung schwierig

Neue Plattform hinzu? -> Neuimplementierung nötig?

# .NET Standard

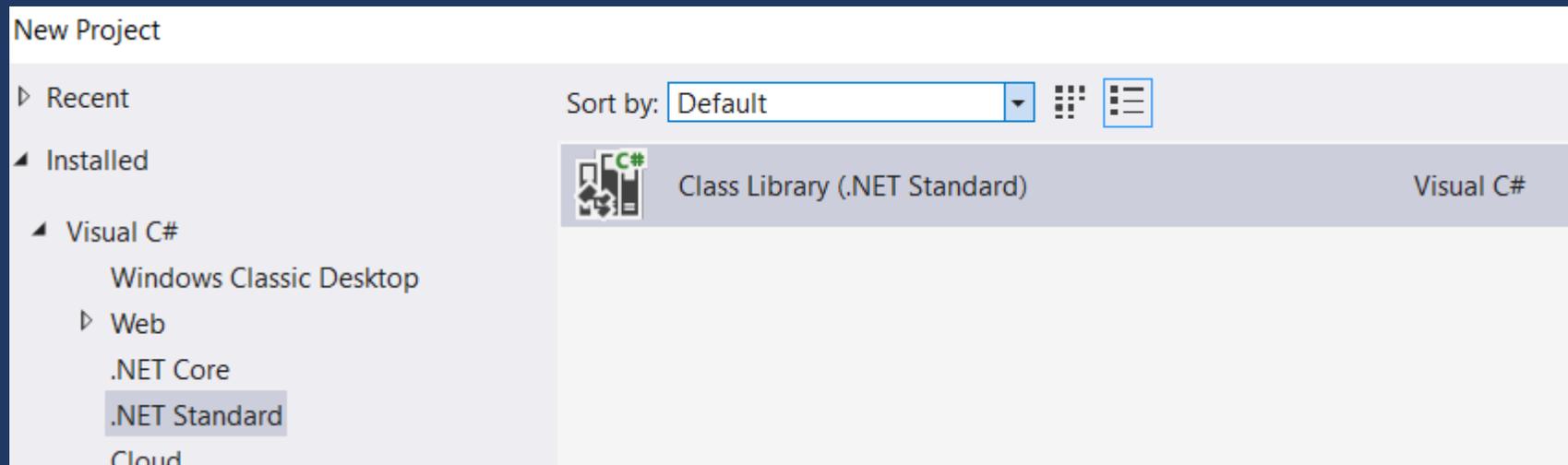




# .NET Standard

.NET Standard ist eine Spezifikation

Set von APIs – muss von allen .NET Plattformen implementiert werden



# .NET Standard

.NET Standard

~

HTML Spezifikation

.NET Framework

.NET Core

Xamarin

~

Browsers

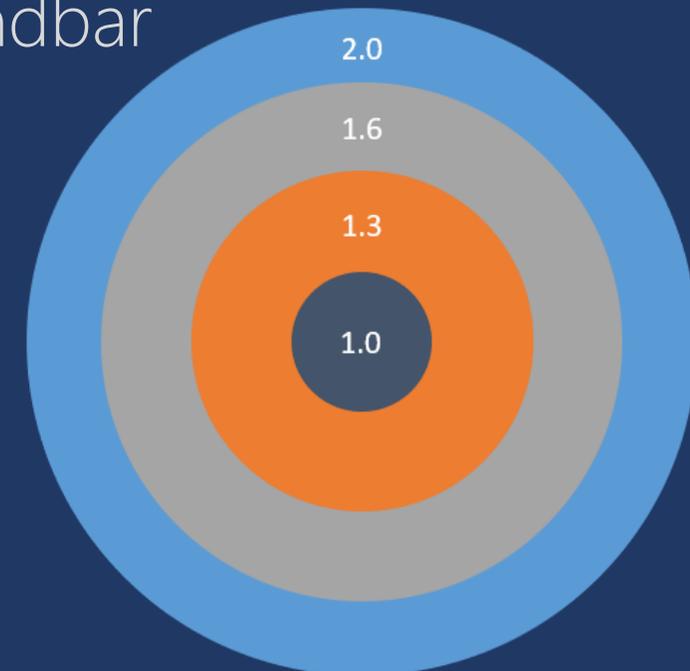
# .NET Standard – Versionierung

Höhere Versionen enthalten alle APIs von niedrigeren Versionen

.NET Standard Projekt 2.0 kann zB 1.3 verwenden

.NET Implementierungen unterstützen konkrete Version

.NET Standard bis zu "dieser Version" verwendbar



# .NET Standard 2.0

Viele APIs hinzugekommen

Hauptsächlich vom .NET Framework ("Full") und Xamarin

.NET Core 2.0 dadurch "größer" (da .NET Standard 2.0 impl.)

Verwendung von .NET Framework ("Full") Libraries in .NET Standard 2.0

Compat Shim (ohne Neukompilierung)

Für APIs, die in .NET Standard 2.0 "enthalten" sind

+20K

More APIs than  
.NET Standard  
1.x

~70%

of NuGet packages  
are API compatible

# Welche Version von .NET Standard verwenden?

Je höher die Version, desto mehr APIs

Je niedriger die Version, desto mehr Plattformen

Niedrigere Version

Höhere Version



Mehr Reichweite

Mehr APIs

# .NET Standard Support

## .NET implementation support

The following table lists all versions of .NET Standard and the platforms supported:

12

| .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 (with .NET Core 1.x SDK) | 4.5  | 4.5  | 4.5.1 | 4.6  | 4.6.1 | 4.6.2      |            |            |
| .NET Framework (with .NET Core 2.0 SDK) | 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        |
| Universal Windows Platform              | 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  |      |       |      |       |            |            |            |

| Target Framework      | Latest Version | Target Framework Moniker (TFM) | Implemented .NET Standard Version |
|-----------------------|----------------|--------------------------------|-----------------------------------|
| .NET Standard         | 2.0            | netstandard2.0                 | N/A                               |
| .NET Core Application | 2.0            | netcoreapp2.0                  | 2.0                               |
| .NET Framework        | 4.7.1          | net471                         | 2.0                               |

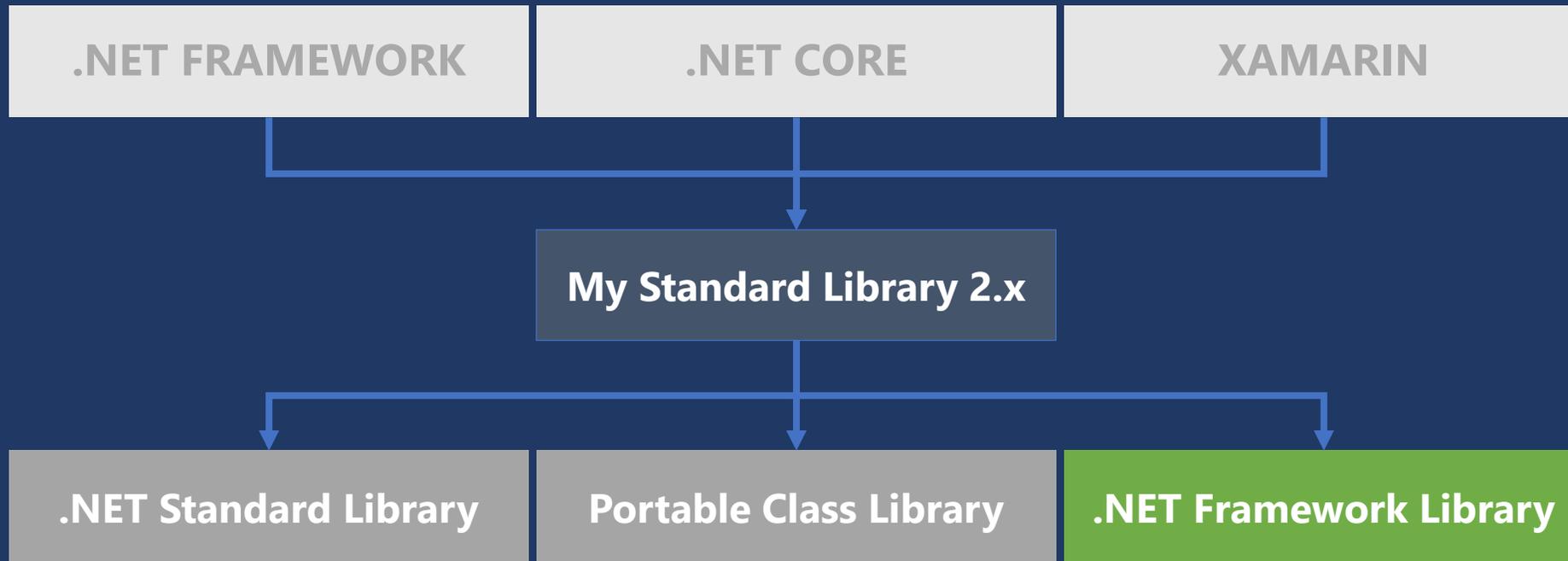
<https://docs.microsoft.com/en-us/dotnet/standard/net-standard#net-implementation-support>

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

<https://github.com/dotnet/standard/blob/master/docs/faq.md>

<https://immo.landwerth.net/netstandard-versions>

# .NET Standard 2.0



**Legend**

→ Is able to reference

Application Type

Via Portability

Via Compatibility Shim

# .NET Standard 2.0 – Übersetzungszeit



# .NET Standard 2.0 – Laufzeit



# Plattform-spezifische APIs

.NET Standard enthält (fast) nur APIs, die "überall" funktionieren

Einige (wenige) APIs werfen *PlatformNotSupportedException*

Windows Compatibility Pack for .NET Core (Microsoft.Windows.Compatibility)

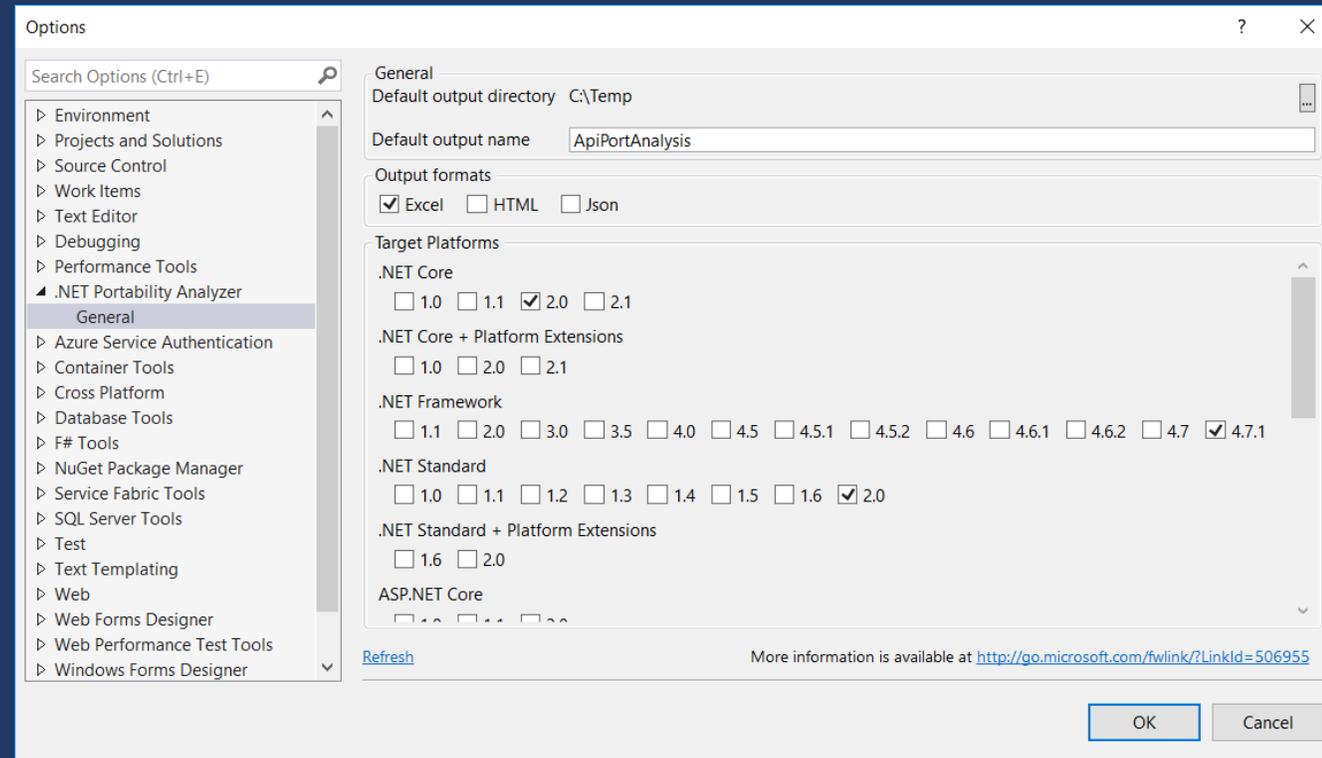
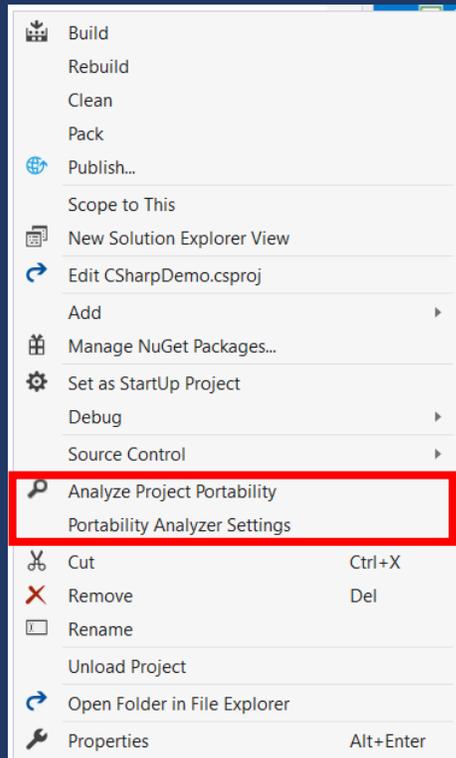
Plattform-spezifische APIs bauen auf .NET Standard auf

Schränkt Portabilität ein

zB Registry, Reflection Emit, Access Control, Windows Identity

# .NET Portability Analyzer

```
ApiPort.exe analyze -f MyLib.dll
```



<https://github.com/Microsoft/dotnet-apiport>

<https://marketplace.visualstudio.com/items?itemName=ConnieYau.NETPortabilityAnalyzer>

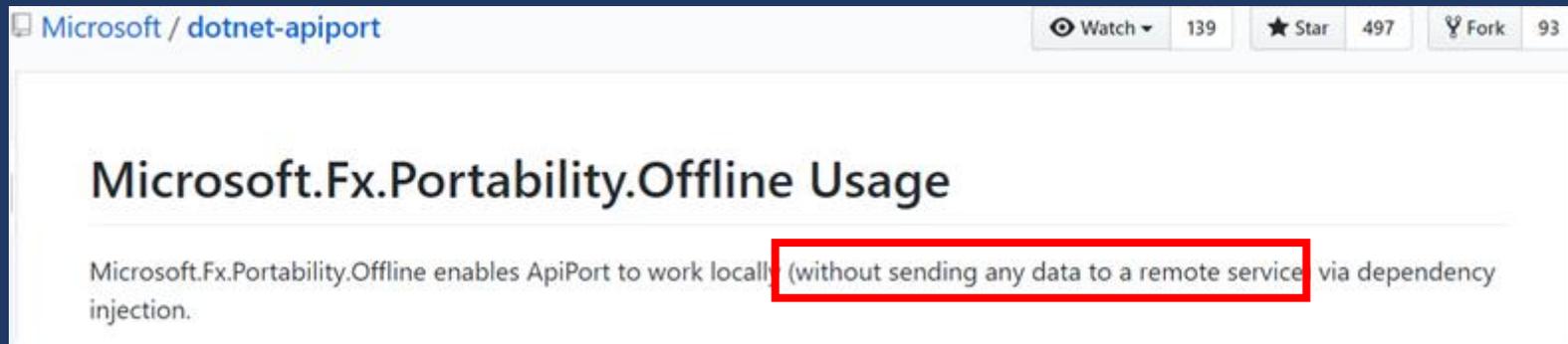
# .NET Portability Analyzer – Ergebnisse

|   |                                  |   |           |                             |                               |                            |                            |
|---|----------------------------------|---|-----------|-----------------------------|-------------------------------|----------------------------|----------------------------|
| 1 | Submission Id                    | 10b4617f-546f-4b93-b3c7-51e823ce3feb  |           |                             |                               |                            |                            |
| 2 | Description                      |   |           |                             |                               |                            |                            |
| 3 | Targets                          | .NET Core,.NET Framework,Version=v4.5,.NET Framework,Version=v4.7.1,.NET Standard,Version=v1.6,.NET Standard,Version=v2.0 |           |                             |                               |                            |                            |
| 4 |                                  |   |           |                             |                               |                            |                            |
| 5 | Header for assembly name entries | Target Framework  | .NET Core | .NET Framework,Version=v4.5 | .NET Framework,Version=v4.7.1 | .NET Standard,Version=v1.6 | .NET Standard,Version=v2.0 |
| 6 | DemoLibrary                      | .NETCoreApp,Version=v2.0  | 100       | 100                         | 100                           | 81.58                      | 100                        |

| Target type               | Target member                                  | Header for  | .NET Core       | .NET Framework,Version=v4.5 | .NET Framework,Version=v4.7.1 | .NET Standard,Version=v1.6 | .NET Standard,Version=v2.0 | Recommended changes                                  |
|---------------------------|--|-------------|-----------------|-----------------------------|-------------------------------|----------------------------|----------------------------|--|
| T:System.Data.DataSet     | T:System.Data.DataSet                          | DemoLibrary | Supported: 2.0+ | Supported: 1.1+             | Supported: 1.1+               | Supported: 2.0+            | Supported: 2.0+            |  |
| T:System.Data.DataSet     | M:System.Data.DataSet.#ctor                    | DemoLibrary | Supported: 2.0+ | Supported: 1.1+             | Supported: 1.1+               | Supported: 2.0+            | Supported: 2.0+            |  |
| T:System.Threading.Thread | T:System.Threading.Thread                      | DemoLibrary | Supported: 1.0+ | Supported: 1.1+             | Supported: 1.1+               | Supported: 2.0+            | Supported: 2.0+            |  |
| T:System.Threading.Thread | M:System.Threading.Thread.get_CurrentThread    | DemoLibrary | Supported: 1.0+ | Supported: 1.1+             | Supported: 1.1+               | Supported: 2.0+            | Supported: 2.0+            |  |
| T:System.Threading.Thread | M:System.Threading.Thread.get_CurrentUICulture | DemoLibrary | Supported: 2.0+ | Supported: 1.1+             | Supported: 1.1+               | Supported: 2.0+            | Supported: 2.0+            | Use System.Globalization.CultureInfo.CurrentCulture. |
| T:System.Drawing.Color    | T:System.Drawing.Color                         | DemoLibrary | Supported: 2.0+ | Supported: 1.1+             | Supported: 1.1+               | Supported: 2.0+            | Supported: 2.0+            |  |
| T:System.Drawing.Color    | M:System.Drawing.Color.get_Blue                | DemoLibrary | Supported: 2.0+ | Supported: 1.1+             | Supported: 1.1+               | Supported: 2.0+            | Supported: 2.0+            |  |

# .NET Portability Analyzer

- Up-to-date server-side analysis
  - **NOTE** .NET platform targets and report formats are constantly fetched from an Azure cloud service that is updated constantly. In addition, the analysis is performed remotely so the information is always the latest.
  - We only **send .NET APIs to the service** to analyze for portability. For more information, check out our [privacy policy](#).



Microsoft / dotnet-apiport

Watch 139 Star 497 Fork 93

## Microsoft.Fx.Portability.Offline Usage

Microsoft.Fx.Portability.Offline enables ApiPort to work locally (without sending any data to a remote service) via dependency injection.

# Platform-Compat Analyzer

dotnet / platform-compat Watch 66 Star 134 Fork 15

Code Issues 24 Pull requests 1 Projects 0 Wiki Insights

Roslyn analyzer that finds usages of APIs that will throw PlatformNotSupportedException on certain platforms.

 **Microsoft.DotNet.Analyzers.Compatibility** by Microsoft v0.2.12-alpha

Prerelease Shows diagnostics for deprecated APIs and .NET Core APIs that throw PlatformNotSupportedException on specific operating systems.

```
var appDomain = AppDomain.CreateDomain("myDomain");
```

Entire Solution 0 Errors 1 Warning 0 Messages Build + IntelliSense

| Code  | Description   |
|-------|---|
| PC001 | AppDomain.CreateDomain(string) isn't supported on Linux, macOS, and Windows |

<https://github.com/dotnet/platform-compat>

# .NET Portability Analyzer / Platform-Compat Analyzer

## .NET Portability Analyzer

| Header for assembly name entries | Target Framework           | .NET Core | .NET Framework | .NET Standard |
|----------------------------------|----------------------------|-----------|----------------|---------------|
| DemoLibrary                      | .NETFramework,Version=v4.7 | 100       | 100            | 100           |

## Platform-Compat Analyzer

| Code  | Description   |
|---|---|
|  PC001 | AppDomain.CreateDomain(string) isn't supported on Linux, macOS, and Windows |

# Wann .NET Core?

.NET Core Features benötigt (Cross-Platform)

Cloud / Microservices / Docker (Linux) / Performance – “Pay-As-You-Go”

Side-by-side .NET Versionen / Self contained deployment

Neues Projekt (wenn Features in .NET Core vorhanden)

CLI (Windows, Mac, Linux)

# Wann NICHT .NET Core => .NET (Full) Framework?

“Alles läuft” / Keine Notwendigkeit - Kein echter “Business Reason”

Features benötigt, die in .NET Core nicht vorhanden sind

- ASP.NET WebForms, ASP.NET WebPages

- WCF (aber Client für .NET Core vorhanden)

- WPF

- WinForms

- WF

# Wann .NET Standard?

Immer ;)

Möglichkeiten ausloten (.NET Portability Analyzer, Platform-Compat Analyzer)

Migration einzelner Libraries

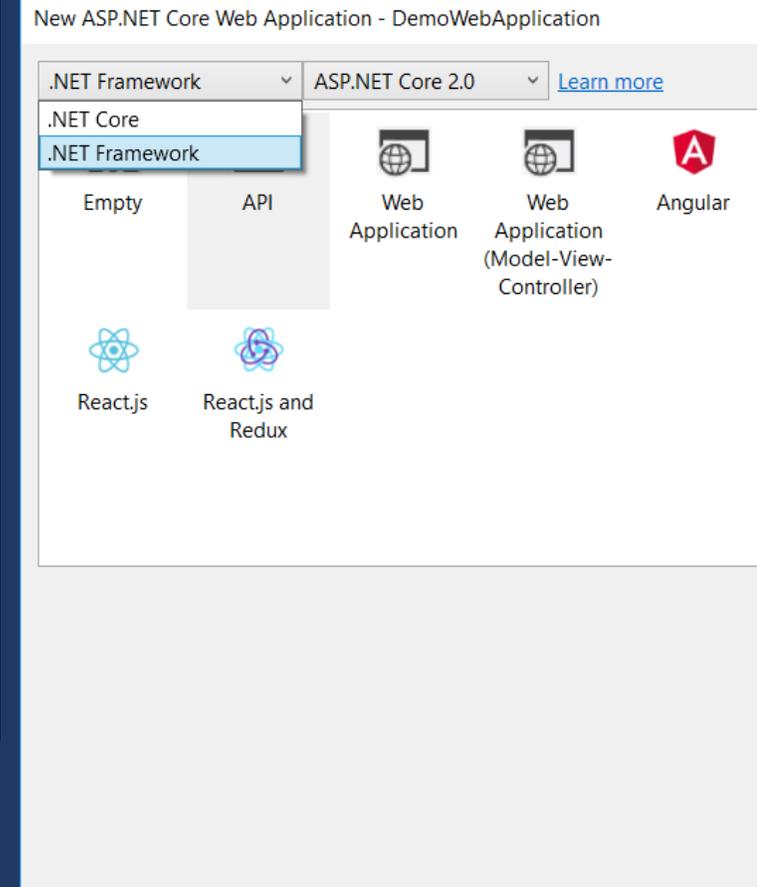
Investition in Zukunft

Neue Libraries / Auswahl Libraries

# ASP.NET Core

## Which one is right for me?

| ASP.NET Core  | ASP.NET  |
|---|--|
| Build for Windows, macOS, or Linux  | Build for Windows  |
| <a href="#">Razor Pages</a> is the recommended approach to create a Web UI as of ASP.NET Core 2.x. See also <a href="#">MVC</a> , <a href="#">Web API</a> , and <a href="#">SignalR</a> . | Use <a href="#">Web Forms</a> , <a href="#">SignalR</a> , <a href="#">MVC</a> , <a href="#">Web API</a> , or <a href="#">Web Pages</a> |
| Multiple versions per machine   | One version per machine  |
| Develop with Visual Studio, <a href="#">Visual Studio for Mac</a> , or <a href="#">Visual Studio Code</a> using C# or F#  | Develop with Visual Studio using C#, VB, or F#   |
| Higher performance than ASP.NET   | Good performance   |
| <a href="#">Choose .NET Framework or .NET Core runtime</a>  | Use .NET Framework runtime   |

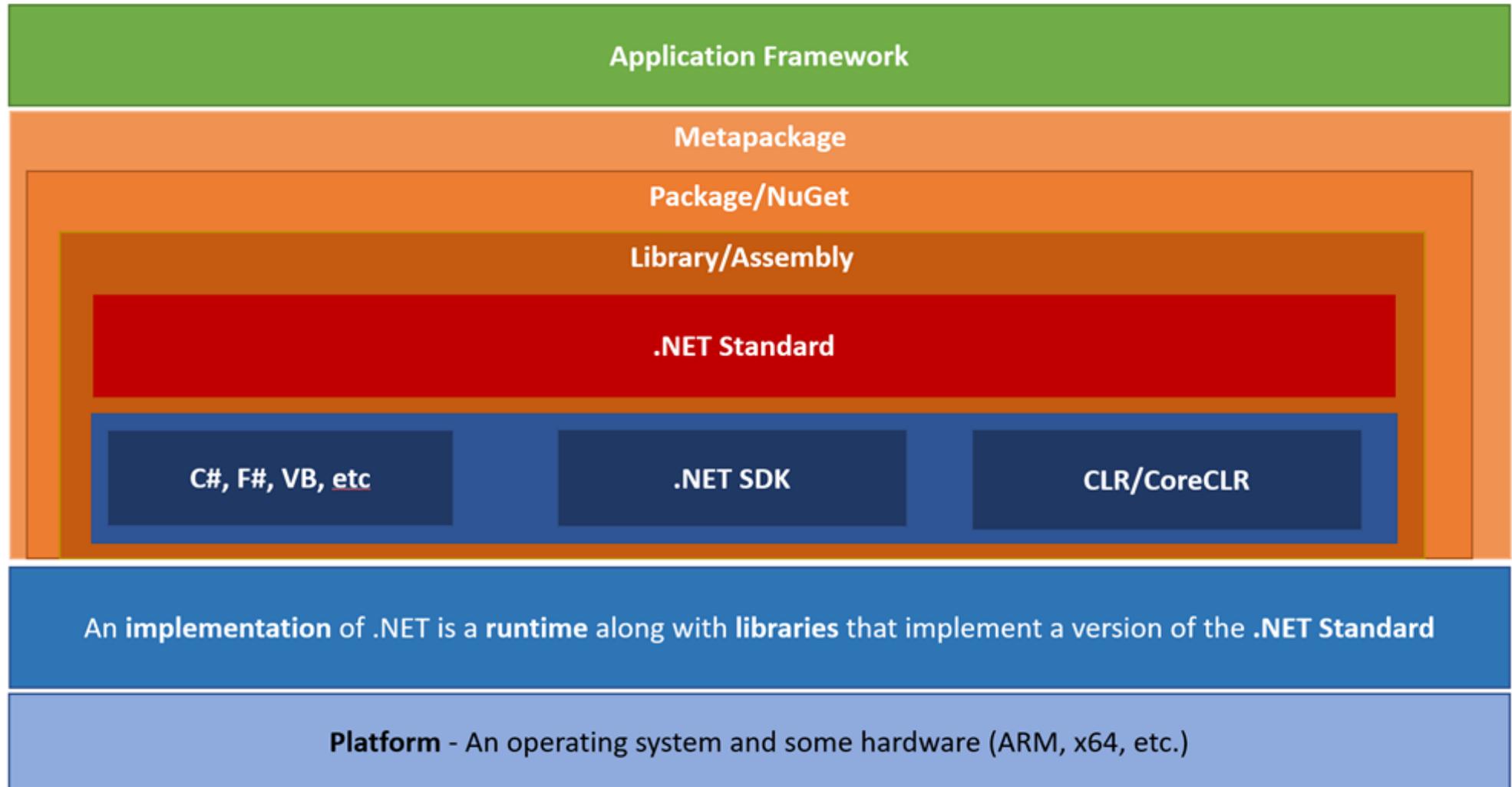


# EF Core

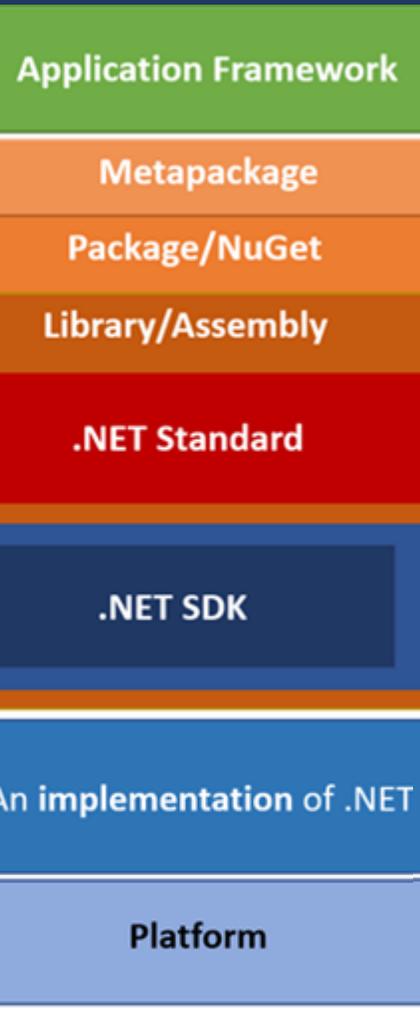
| .NET implementation   | Status   | EF Core 1.x requirements   | EF Core 2.x requirements <sup>(1)</sup>                                 |
|---|--|--|---|
| <b>.NET Core</b> (ASP.NET Core, Console, etc.)                | Fully supported and recommended                                    | .NET Core SDK 1.x  | .NET Core SDK 2.x   |
| <b>.NET Framework</b> (WinForms, WPF, ASP.NET, Console, etc.) | Fully supported and recommended. EF6 also available <sup>(2)</sup> | .NET Framework 4.5.1   | .NET Framework 4.6.1  |
| <b>Mono &amp; Xamarin</b>                                     | In progress <sup>(3)</sup>   | Mono 4.6<br>Xamarin.iOS 10<br>Xamarin.Mac 3<br>Xamarin.Android 7 | Mono 5.4<br>Xamarin.iOS 10.14<br>Xamarin.Mac 3.8<br>Xamarin.Android 7.5 |
| <b>Universal Windows Platform</b>                             | EF Core 2.0.1 recommended <sup>(4)</sup>                           | .NET Core UWP 5.x package  | .NET Core UWP 6.x package   |

# Do you know .NET?

## .NET - "Do you know .NET?"



# Do you know .NET?



**Application Framework** *Are you using the ASP.NET Core web framework for that microservice?*

**Metapackage** *I want to install the ASP.NET Core framework; it's a package of packages*

**Package/NuGet** *I know there's a NuGet package for decoding JSON.*

**Library/Assembly** *Now, you'll compile your source into an assembly*

**.NET Standard** *Which version of the .NET Standard specification does your assembly target?  
My Apple Watch supports .NET Standard 1.6 but my Windows 10 laptop supports 2.0 with more APIs.*

**C#, F#, VB, etc** *Which language did you use?*

**.NET SDK** *Did you get the developer tools?*

**CLR/CoreCLR** *Which **runtime** is your app using?*

*An **implementation** of .NET is a **runtime** along with **libraries** that implement a version of the .NET Standard*

*Are you using .NET Core, .NET Framework, or Mono for this project?*

**Platform** *An operating system and some hardware (ARM, x64, etc.)*

*Is that an ASP.NET Core app running in Docker on a Raspberry Pi?*



# C# 7.0

*out* Variables

Tuples

Discards

Pattern Matching

*ref locals* and *returns*

Local Functions

More Expression-bodied Members

*throw* Expressions

Generalized *async* return types

Numeric Literal Syntax Improvements

# out Variables

```
if (int.TryParse(input, out int result))
{
    Console.WriteLine(result);
}
else
{
    Console.WriteLine("Could not parse input");
}
```

# Tuples

```
(string, int) GetPerson()  
{  
    return ("Christian", 4061);  
}
```

```
(string Name, int Zip) person = GetPerson();  
Console.WriteLine($"{person.Name} {person.Zip}");
```

```
(var name1, var zip1) = GetPerson();
```

```
var (name2, zip2) = GetPerson();
```

```
(var name3, _) = GetPerson();
```

# Tuples

```
(string Name, int Zip) GetPerson2()  
{  
    return ("Christian", 4061);  
}
```

```
var person = GetPerson2();  
Console.WriteLine($"{person.Name} {person.Zip}");
```

```
var (name, zip) = GetPerson2();
```

# Pattern Matching

## Const Pattern

- o `is null`
- o `is "Chris"`

## Type Pattern

- o `is string`

## Var Pattern

- o `is string s`
- o `is string s && s.Length > 2`

# Pattern Matching

```
object obj1 = "Christian";  
if (obj1 is string s && s.Length > 3)  
    Console.WriteLine(s);
```

```
object obj2 = new Person("Christian", 4061);  
switch (obj2)  
{  
    case Person p when p.Zip > 4000:  
        Console.WriteLine($"{p.Name} {p.Zip} is greater than 4000"); break;  
    case Person p:  
        Console.WriteLine($"{p.Name} {p.Zip}"); break;  
    case null:  
        Console.WriteLine("null"); break;  
}
```

# Local Functions

```
public void Calc()
{
    var x = 20;
    var result = Add(2);

    Console.WriteLine(x);           // 40
    Console.WriteLine(result);     // 42

    int Add(int y)
    {
        x = 40;
        return x + y;
    }
}
```

# Iterator Method (ohne Local Function)

```
public IEnumerable<int> GetValues(int startIndex)
{
    if (startIndex < 0 || startIndex >= values.Length) { throw ... }
    for (var idx = startIndex; idx < values.Length; idx++)
        yield return values[idx];
}
```

```
var result = GetValues(-2);
foreach (var val in result) { ... } 
```

# Iterator Method (mit Local Function)

```
public IEnumerable<int> GetValues(int startIndex)
{
    if (startIndex < 0 || startIndex >= values.Length) { throw ... }
    return GetValuesLocal();

    IEnumerable<int> GetValuesLocal()
    {
        for (var idx = startIndex; idx < values.Length; idx++)
            yield return values[idx];
    }
}

var result = GetValues(-2);
foreach (var val in result) { ... }
```

A red starburst graphic is positioned over the code line 'var result = GetValues(-2);', highlighting the negative index value -2.

## async Method (ohne Local Function)

```
public async Task<int> AddAsync(int val1, int val2)
{
    if (val1 < 0 || val2 < 0) { throw ... }
    await Task.Delay(2000);
    return val1 + val2;
}
```

```
var task = AddAsync(1, -2);
var result = await task;
```



# async Method (mit Local Function)

```
public Task<int> Add(int val1, int val2)
{
    if (val1 < 0 || val2 < 0) { throw ... }
    return AddAsync();

    async Task<int> AddAsync()
    {
        await Task.Delay(2000);
        return val1 + val2;
    }
}
```

```
var task = Add(1, -2);
var result = await task;
```



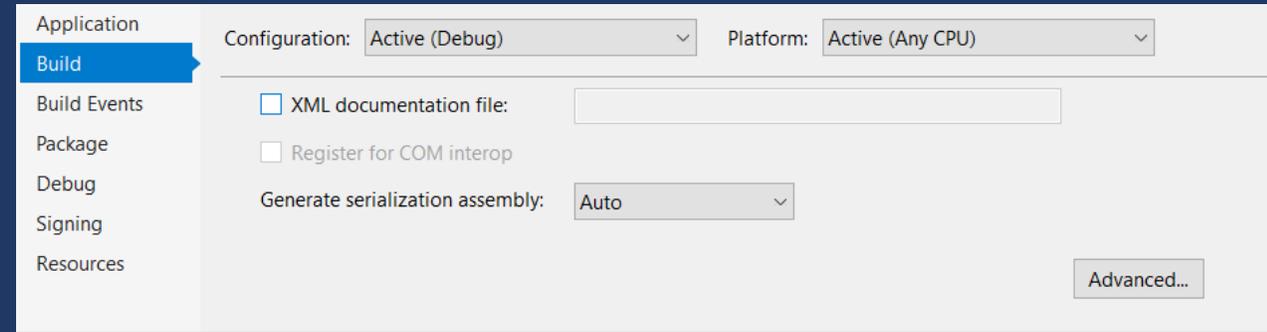
# C# 7.1 & 7.2

## C# 7.1

*async* Main

*default* Literal Expressions

Inferred Tuple Element Names



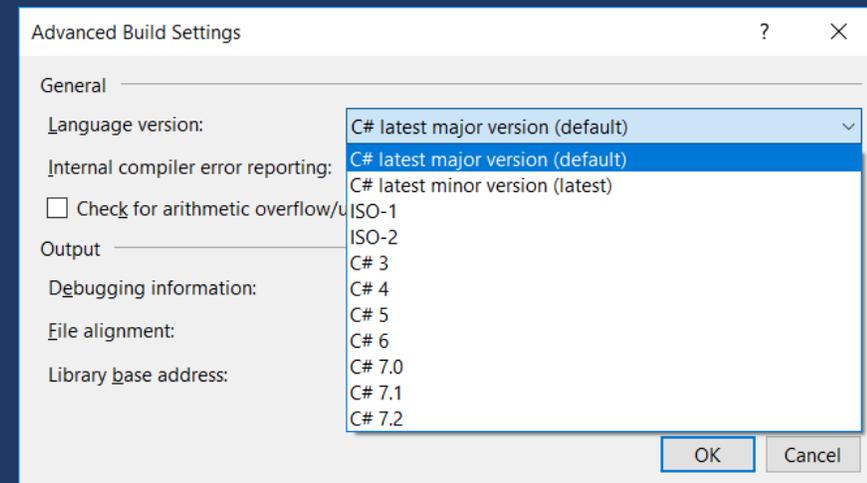
## C# 7.2

Reference Semantics with Value Types

Non-trailing Named Arguments

Leading Underscores in Numeric Literals

*private protected* Access Modifier



# Visual Studio 2017

---

Einige interessante Features

# Visual Studio Familie



## **Visual Studio**

A fully-featured, integrated development environment (IDE) for developing .NET apps on a Windows PC development machine.



## **Visual Studio Code**

Open source, cross-platform editor with .NET support.



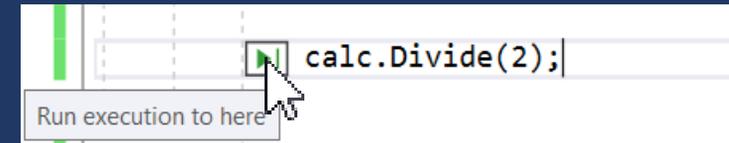
## **Visual Studio for Mac**

A fully-featured IDE for developing .NET apps on a Mac OS development machine.

# Structure Guide Lines / Run to here

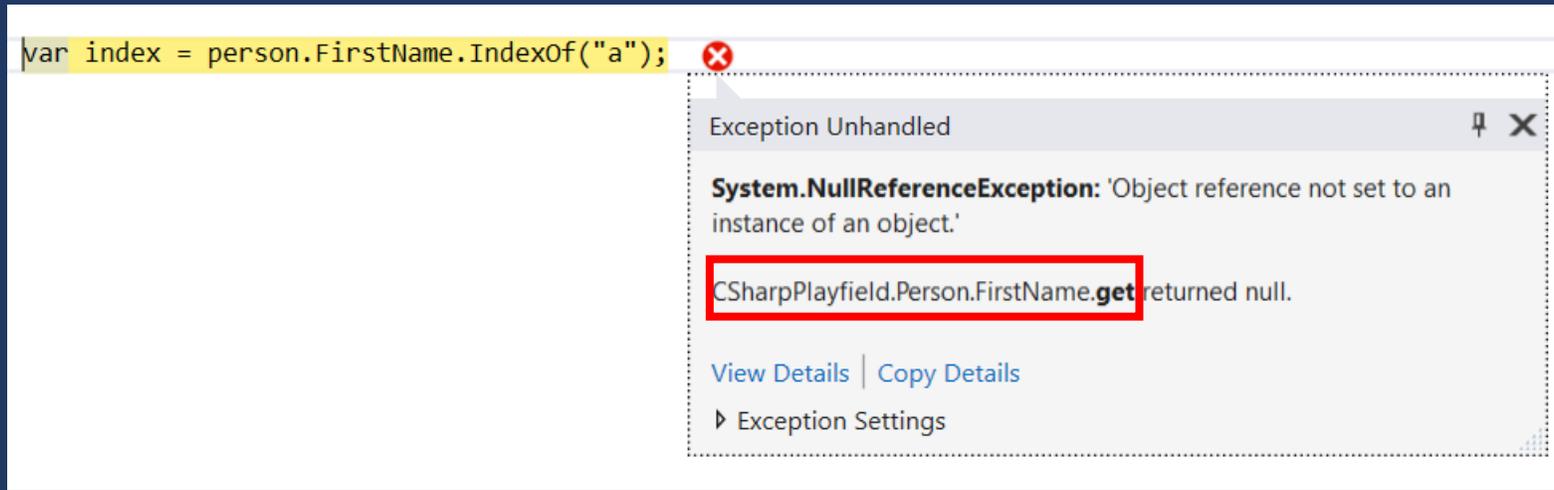
```
5 namespace XUnitTestProject1
6 {
7     0 references | 0 changes | 0 authors, 0 changes
8     public class UnitTest1
9     {
10        [Fact]
11        0 references | 0 changes | 0 authors, 0 changes
12        public void Test1()
13        {
14            var calc = new Calculator
15            {
16                Value = 2
17            };
18            calc.Divide(2);
19            Assert.Equal(1, calc.Value);
20        }
21    }
22 }
```

namespace XUnitTestProject1  
public class UnitTest1  
public void Test1()



# Exception Helper

```
var index = person.FirstName.IndexOf("a");
```



Exception Unhandled

**System.NullReferenceException:** 'Object reference not set to an instance of an object.'

CSharpPlayfield.Person.FirstName.get returned null.

[View Details](#) | [Copy Details](#)

▶ Exception Settings

# Add Reference

The screenshot shows a Visual Studio code editor window. The code in the editor is:

```
11 var calc = new Calculator();
```

The word `Calculator` is underlined with a red squiggly line, indicating a compilation error. A context menu is open over the code, with the option "Add reference to 'UnitTestDemo'." highlighted in blue and enclosed in a red rectangular box. To the right of the code editor, a tooltip displays the error message:

**CS0246** The type or namespace name 'Calculator' could not be found (are you missing a using directive or an assembly reference?)

Below the error message, a list of using directives is shown:

```
using System;  
using UnitTestDemo;  
using Xunit;  
...
```

The `using UnitTestDemo;` line is highlighted in green. At the bottom of the tooltip, the text "Adding reference 'UnitTestDemo' to 'XUnitTestProject1'" is displayed, along with a "Preview changes" link.

...

```
12 var calc = new Calculator();  
13 calc.Value = 2;
```

12 var calc = new Calculator();  
13 calc.Value = 2;

Object initialization can be simplified

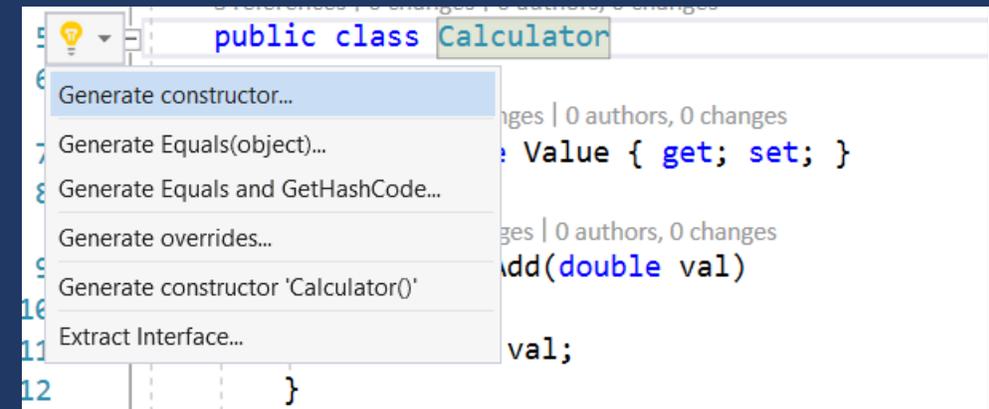
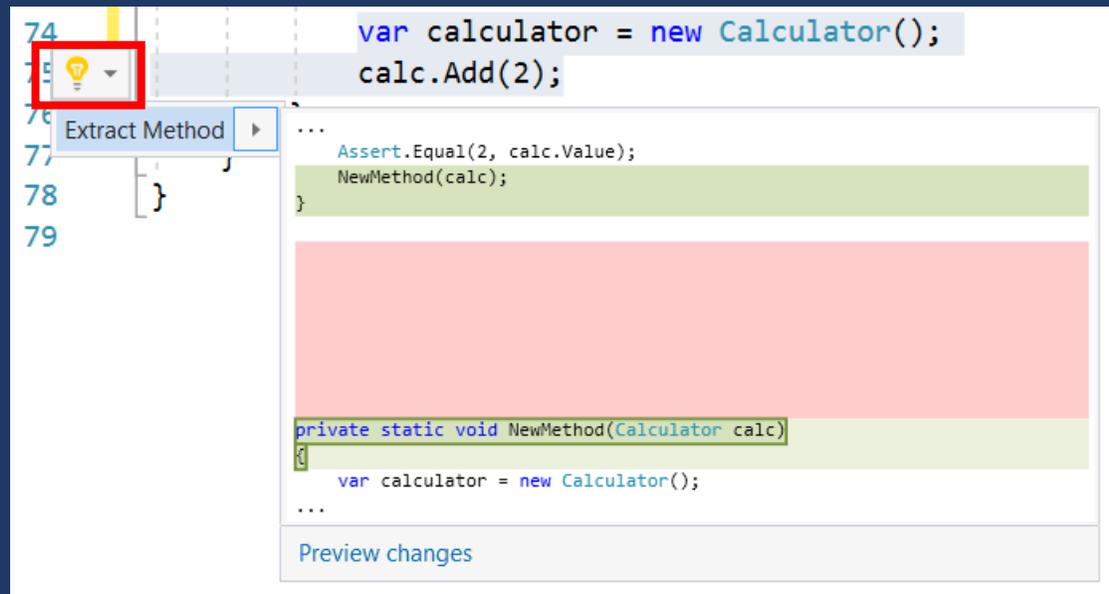
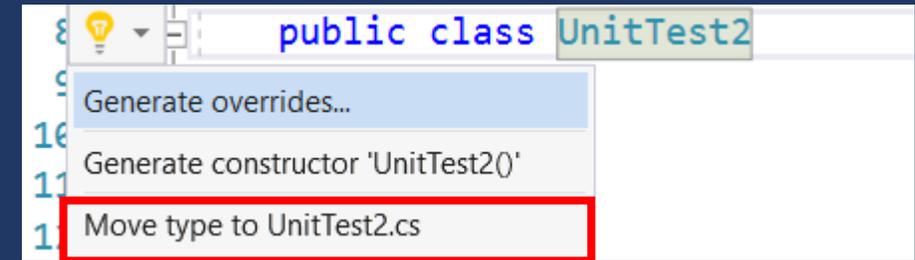
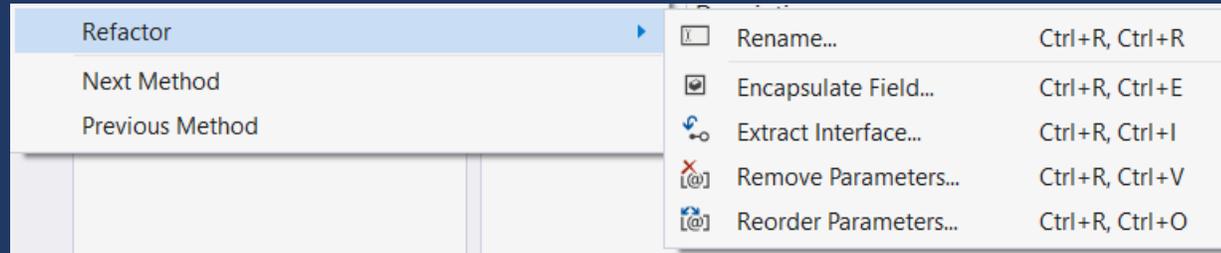
IDE0017 Object initialization can be simplified

Suppress IDE0017

```
var calc = new Calculator();  
calc.Value = 2;  
var calc = new Calculator  
{  
    Value = 2  
};
```

Preview changes  
Fix all occurrences in: Document | Project | Solution

# Refactor / Generate



# Go To All – Ctrl+, Ctrl+T / Code Completion

The screenshot shows four instances of the 'Go to All' search window in Visual Studio, each with a different search term and results. The search terms and results are:

- Go to All:** Search term: `|` (empty). No results are shown.
- Go to Types:** Search term: `tP|`. Results: **Person** (project CSharpPlayfield) and **Program** (project CSharpPlayfield).
- Go to Members:** Search term: `m Fi`. Results: **FirstName** (in Person (project CSharpPlayfield)) and **firstName** (in Person (project CSharpPlayfield)).
- Go to Files:** Search term: `f Un`. Results: **UnitTest1.cs** (c:\Data\Dev\repos\SNEK2018\demo\UnitTestDemo\XUnitTestProject1) and **Calculator.cs** (c:\Data\Dev\repos\SNEK2018\demo\UnitTestDemo\UnitTestDemo).

The screenshot shows a code completion dropdown menu for the text `var calculator = new Ca`. The dropdown lists several options, with **Calculator** selected. The options are:

- AsyncCallback
- Calculator**
- CannotUnloadAppDomainException
- ConsoleCancelEventArgs
- ConsoleCancelEventHandler
- InvalidCastException
- OperationCanceledException
- TestCaseOrdererAttribute

The screenshot shows the same code completion dropdown menu for the text `var calculator = new Ca`. The dropdown lists several options, with **Calculator** selected. The options are:

- AsyncCallback
- Calculator**
- CannotUnloadAppDomainException
- ConsoleCancelEventArgs
- InvalidCastException
- OperationCanceledException
- TestCaseOrdererAttribute

A red box highlights the bottom toolbar of the dropdown menu, which contains icons for navigation and editing.

# .editorconfig

```
[*.cs]
insert_final_newline = true
indent_style = space
indent_size = 4

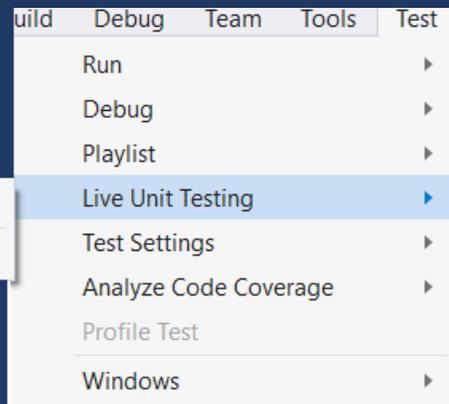
csharp_new_line_before_open_brace = all
csharp_new_line_before_else = true
csharp_new_line_before_catch = true

dotnet_style_qualification_for_field = true:warning

csharp_style_var_when_type_is_apparent = true:warning
```

<https://docs.microsoft.com/en-us/visualstudio/ide/editorconfig-code-style-settings-reference>

# Live Unit Testing (Enterprise)



```
public class Calculator
{
    5 references | 0 changes | 0 authors, 0 changes
    public double Value { get; set; }

    1 reference | 0 changes | 0 authors, 0 changes
    public void Add(double val)
    {
        Value += val;
    }

    1 reference | 0 changes | 0 authors, 0 changes
    public void Divide(double divisor)
    {
        if (divisor == 0)
        {
            throw new ArgumentException($"{nameof(divisor)} can't be 0");
        }
    }

    Value *= divisor;
}
}
```



Christian Schwendtner  
PROGRAMMIERFABRIK GmbH

 @CSchwendtner



# Quellen – Grafiken / Weiterführende Links

<https://github.com/dotnet-presentations>

<https://github.com/dotnet-presentations/home>

<https://github.com/dotnet-presentations/dotnetcore-workshop>

<https://github.com/dotnet-presentations/aspnetcore-app-workshop>

<https://dotnet.github.io/>

<https://docs.microsoft.com/en-us/dotnet/core/>

<https://github.com/dotnet/core>

<https://github.com/dotnet/standard>

<https://github.com/dotnet/standard/tree/master/docs/netstandard-20>

<https://github.com/dotnet/standard/blob/master/docs/faq.md>

<https://docs.microsoft.com/en-us/dotnet/standard/net-standard#net-implementation-support>

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

<https://github.com/dotnet/standard/blob/master/docs/faq.md>

<https://immo.landwerth.net/netstandard-versions>

<https://docs.microsoft.com/en-us/dotnet/core/packages>