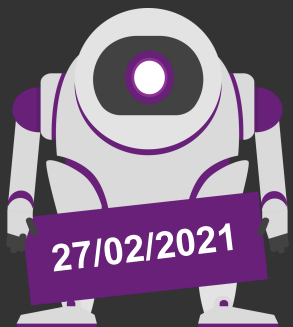# Netcoreconf

What is new in .NET 5
and the future of .NET

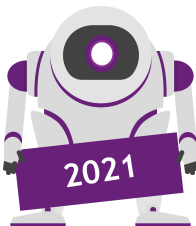**Johnny Hooyberghs**

Passionate Developer @ Involved

@djohnnieke

27/02/2021

SPONSORS

What is new in .NET 5 and the future of .NET
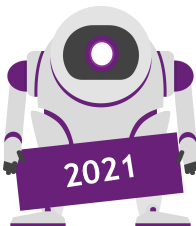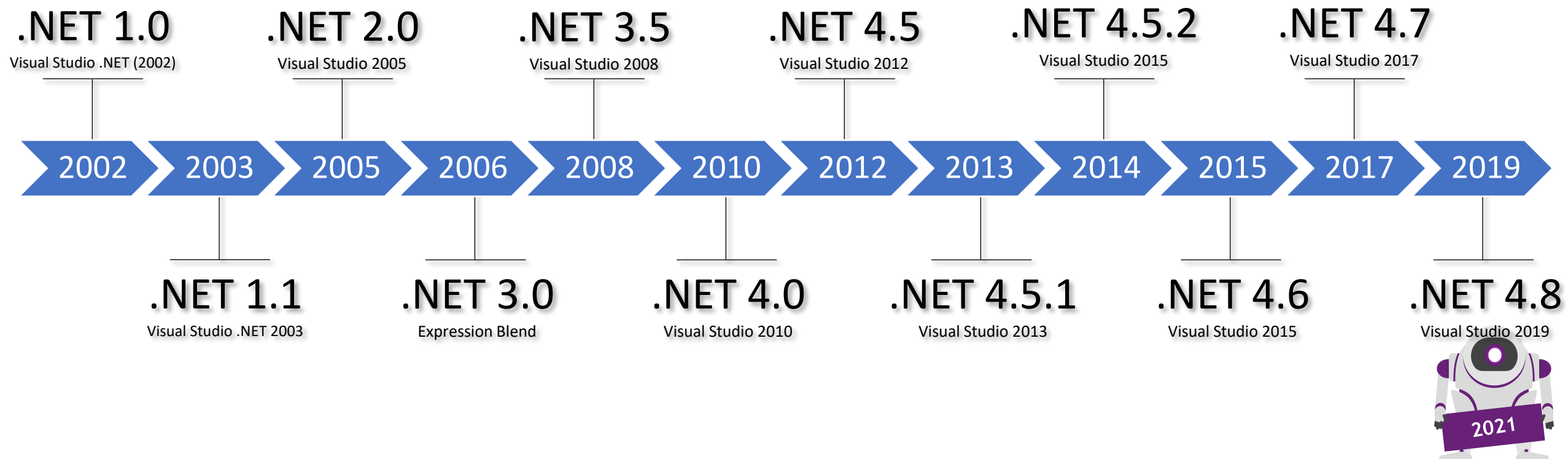
Johnny Hooyberghs

Microsoft® Most Valuable Professional

@djohnnieke

github.com/Djohnnie

www.involved-it.be

johnny.hooyberghs@involved-it.be

2021

involved

https://github.com/Djohnnie/DotNet5-NetCoreConf-2021

#netcoreconf

.NET: a quick history

.NET 1.0
Visual Studio .NET (2002)

.NET 2.0
Visual Studio 2005

.NET 3.5
Visual Studio 2008

.NET 4.5
Visual Studio 2012

.NET 4.5.2
Visual Studio 2015

.NET 4.7
Visual Studio 2017

| 2002 | 2003 | 2005 | 2006 | 2008 | 2010 | 2012 | 2013 | 2014 | 2015 | 2017 | 2019 |

.NET 1.1
Visual Studio .NET 2003

.NET 3.0
Expression Blend

.NET 4.0
Visual Studio 2010

.NET 4.5.1
Visual Studio 2013

.NET 4.6
Visual Studio 2015

.NET 4.8
Visual Studio 2019

involved

https://github.com/Djohnnie/DotNet5-NetCoreConf-2021

#netcoreconf

2021

.NET Core: a quick history

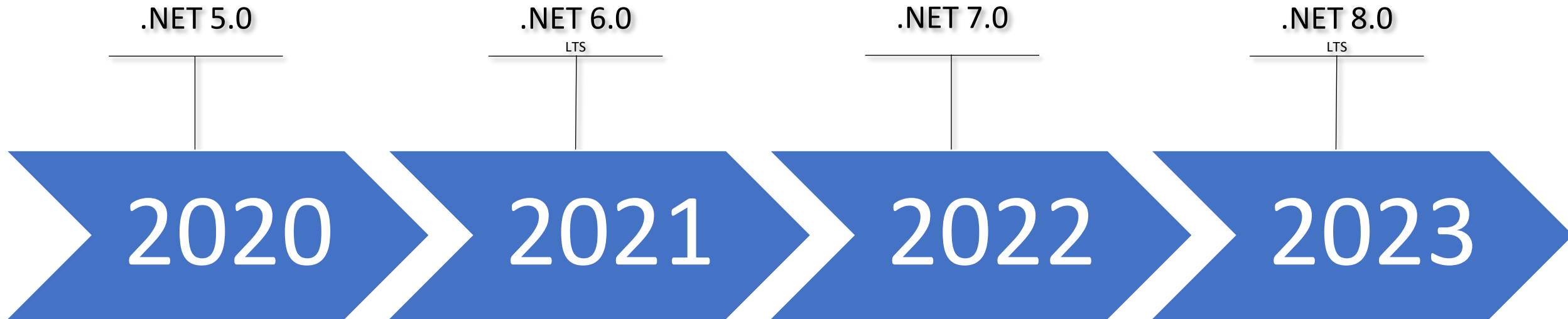| .NET Core 1.0	.NET Core 1.1	.NET Core 2.0	.NET Core 2.1	.NET Core 2.2	.NET Core 3.0	.NET Core 3.1
Visual Studio 2015 Update 3	Visual Studio 2017 Version 15.0	Visual Studio 2017 Version 15.3	Visual Studio 2017 Version 15.7	Visual Studio 2019 Version 16.0	Visual Studio 2019 Version 16.3	Visual Studio 2019 Version 16.4
2016	2016	2017	2018	2018	2019	2019

.NET Core 1.0	June 27, 2019
.NET Core 1.1	June 27, 2019
.NET Core 2.0	October 1, 2018
.NET Core 2.1	August 21, 2021
.NET Core 2.2	December 23, 2019
.NET Core 3.0	March 3, 2020
.NET Core 3.1	December 3, 2022

involved   https://github.com/Djohnnie/DotNet5-NetCoreConf-2021   #netcoreconf

2021
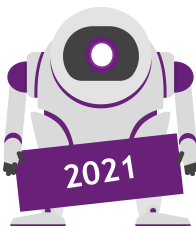
.NET: what about the future?

.NET 5.0	.NET 6.0 LTS	.NET 7.0	.NET 8.0 LTS

2020 **2021** **2022** **2023**

.NET 5	3 months after .NET 6 release (around February 2022)
.NET LTS	Three years after initial release

2021
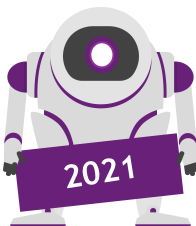
.NET as multi-platform

- CLR, Mono, …

- Portable Class Libraries

- .NET Standard

- .NET 5

2021

.NET = A unified platform

https://devblogs.microsoft.com/dotnet/introducing-net-5/
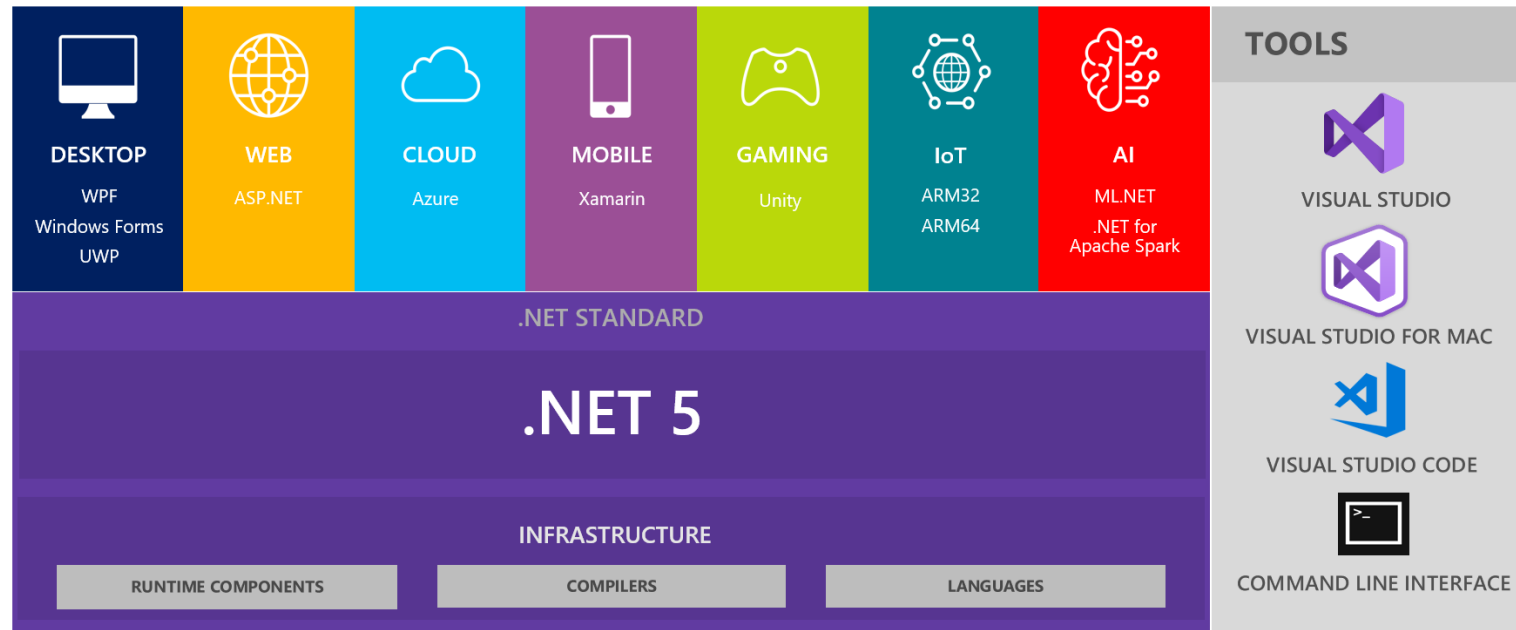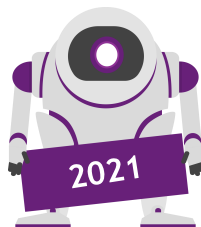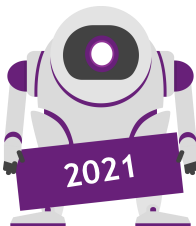
.NET 5 = .NET Core vNext

- Single runtime and BCL
- Unified developer experience
- Expand .NET with capabilities of .NET Core, .NET Framework, Xamarin and Mono
- Single open-source code base

2021

.NET Framework, .NET Core or .NET 5?

- .NET Framework will NOT get new features but will stay supported with every new version of Windows (.NET 4.8 is part of Windows)

- .NET Core 3.1 is LTS (12/2022) but will NOT get new features

- .NET Standard 2.1 will be the last version

- .NET 5 supports .NET Standard 2.1

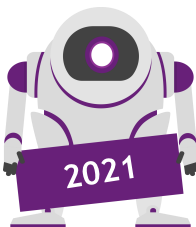- .NET Standard 2.0 is the last supported by .NET Framework
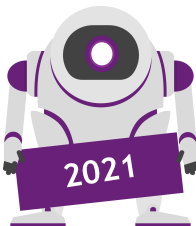
2021
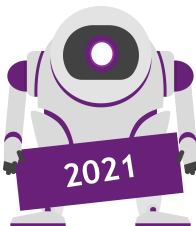
THIS DEMO USES

"PRODUCTION-READY CODE"

Example 1 – Single SDK Experience

- Single CLI

- Single *.csproj format

- Target framework names (TFM)

- "net5.0" replaces and combines "netcoreapp" and "netstandard"

- "net5.0-windows" is OS-specific and includes OS-specific bindings
    - "net6.0-android", "net6.0-ios"

2021

Example 1 – Single SDK Experience

```xml
<PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net5.0</TargetFramework>
    <RootNamespace>_01_SingleSdkExperience.DotNet5</RootNamespace>
</PropertyGroup>
```
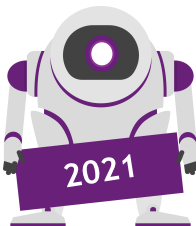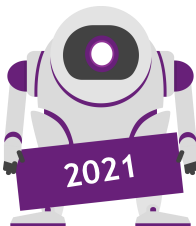
2021

Example 1 – Single SDK Experience

```xml
<PropertyGroup>

    <OutputType>WinExe</OutputType>

    <TargetFramework>net5.0-windows</TargetFramework>

    <RootNamespace>_01_SingleSdkExperience.WinForms</RootNamespace>

    <UseWindowsForms>true</UseWindowsForms>

</PropertyGroup>
```
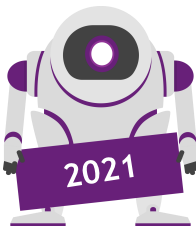
2021

Example 1 – Single SDK Experience

```xml
<PropertyGroup>

    <OutputType>WinExe</OutputType>

    <TargetFramework>net5.0-windows</TargetFramework>

    <RootNamespace>_01_SingleSdkExperience.Wpf</RootNamespace>

    <UseWPF>true</UseWPF>

</PropertyGroup>
```

2021

Example 2 – Default Executables

- Since .NET Core 3.0

- <OutputType>Exe</OutputType> for executable project templates

- Generates *.exe (wrapper) next to .NET DLL

```xml
<PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net5.0</TargetFramework>
    <RootNamespace>_02_DefaultExecutables</RootNamespace>
</PropertyGroup>
```

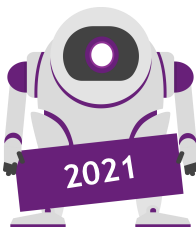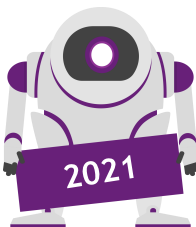# Example 3 – Single file applications

- Since .NET Core 3.0

- Publish feature

- Includes runtime and dependencies

- Platform SDK decides how to build native executable

```
<RuntimeIdentifier>win10-x64</RuntimeIdentifier>
<PublishSingleFile>true</PublishSingleFile>
```

2021

Example 3 – Single file applications

```xml
<PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net5.0</TargetFramework>
    <RootNamespace>_03_SingleFileApplications</RootNamespace>
    <RuntimeIdentifier>win10-x64</RuntimeIdentifier>
    <PublishSingleFile>true</PublishSingleFile>
</PropertyGroup>
```
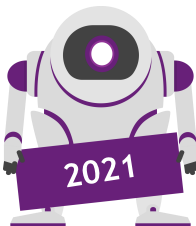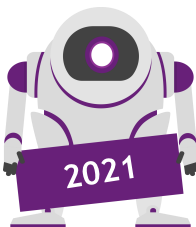
2021

Example 4 – Assembly Linking

- Since .NET Core 3.0

- Publish feature

- Includes runtime and dependencies

- Strips unused dependencies as much as possible

- Platform SDK decides how to build native executable

```xml
<RuntimeIdentifier>win10-x64</RuntimeIdentifier>
<PublishSingleFile>true</PublishSingleFile>
<PublishTrimmed>true</PublishTrimmed>
```

involved

2021

Example 4 – Assembly Linking

```xml
<PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net5.0</TargetFramework>
    <RootNamespace>_04_AssemblyLinking</RootNamespace>
    <RuntimeIdentifier>win10-x64</RuntimeIdentifier>
    <PublishSingleFile>true</PublishSingleFile>
    <PublishTrimmed>true</PublishTrimmed>
</PropertyGroup>
```
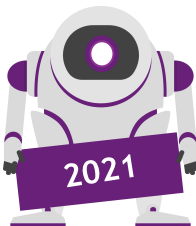
Example 5 – Hardware Intrinsics

- Since .NET Core 1.0 (SIMD, System.Numerics)
- Since .NET Core 3.0 (System.Runtime.Intrinsics)
- Since .NET 5 (support for ARM)
- SIMD has fallback if not supported on CPU
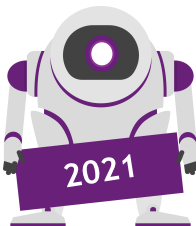- Intrinsics does NOT have fallback if not supported on CPU

```csharp
using System.Numerics;
using System.Runtime.Intrinsics;
using System.Runtime.Intrinsics.Arm;
using System.Runtime.Intrinsics.X86;
```

involved

#netcoreconf

2021

Example 5 – Hardware Intrinsics

```csharp
while (i < lastBlockIndex)
{
    resultVector = Sse2.Add(resultVector, Sse2.LoadVector128(sourcePointer + i));
    i += 4;
}
resultVector = Ssse3.HorizontalAdd(resultVector, resultVector);
resultVector = Ssse3.HorizontalAdd(resultVector, resultVector);

while (i < lastBlockIndex)
{
    resultVector = AdvSimd.Add(resultVector, AdvSimd.LoadVector128(sourcePointer + i));
    i += 4;
}
resultVector = AdvSimd.Add(resultVector, resultVector);
resultVector = AdvSimd.Add(resultVector, resultVector);
```
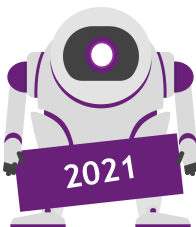
Example 6 – New dev templates

- Since .NET Core 2.0 and 3.0, also supported in .NET 5

- Worker Service (Comparable: Windows Service in .NET Framework)

- gRPC Service (Comparable: WCF Service in .NET Framework)

- WPF template
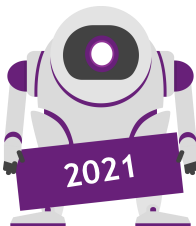
- WinForms template

2021

Example 7 – C# 9.0 features

- Top-level programs

- Target typed new()-expressions

- Init-only properties

- Records

- (No language support for the new System.Half type)

involved

2021

#netcoreconf
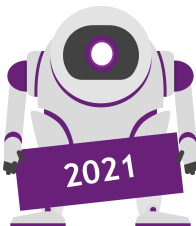
Example 7 – C# 9.0 – Top-level

```csharp
using System;
using System.Runtime.InteropServices;


Console.WriteLine("Hello World!");
FromWhom();
Show.Excitement("Top-level programs can be brief", 8);
```
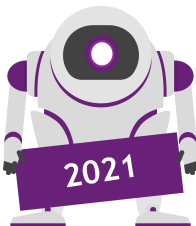
Example 7 – C# 9.0 – new()

```csharp
using System.Collections.Generic;

Dictionary<string, string> dictionary1 = new Dictionary<string, string>();
var dictionary2 = new Dictionary<string, string>();
Dictionary<string, string> dictionary3 = new();
```
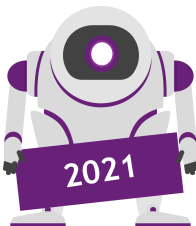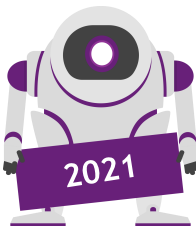
Example 7 – C# 9.0 – init

```csharp
public class Person
{
    public string Name { get; init; }
    public string FirstName { get; init; }
}
```
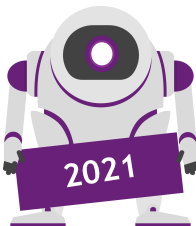
Example 7 – C# 9.0 – records

```csharp
public record Person(string Name, string FirstName);


Person person1 = new("Hooyberghs", "Johnny");
Person person2 = person1 with { FirstName = "Marina" };
```
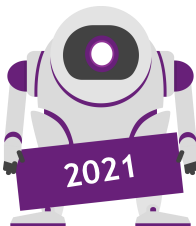
Example 7 – C# 9.0 – Half

```csharp
Half half1 = 0;
Half half2 = 0.0;
Half half = (Half)0;
float f = (float)half;
Half h = (Half)f;
```

Example 8 – Windows Api's

- Windows API's are automatically available based on the TFM

- TFM net5.0-windows including minimum supported version

- net5.0-windows10.0.17763.0

```
<PropertyGroup>
    <OutputType>WinExe</OutputType>
    <TargetFramework>net5.0-windows10.0.17763.0</TargetFramework>
    <RootNamespace>_08_WindowsApis.WinForms</RootNamespace>
    <UseWindowsForms>true</UseWindowsForms>
</PropertyGroup>
```
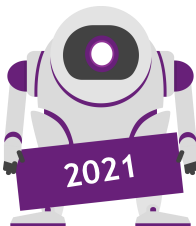
2021

Example 8 – Windows Api's

```csharp
private async void button1_Click(object sender, EventArgs e)
{
    // Initialize the webcam
    MediaCapture captureManager = new MediaCapture();
    await captureManager.InitializeAsync();


    ImageEncodingProperties imgFormat = ImageEncodingProperties.CreateJpeg();
    // create storage file in local app storage
    StorageFile file = await KnownFolders.CameraRoll.CreateFileAsync("TestPhoto.jpg",
                                    CreationCollisionOption.GenerateUniqueName);


    // take photo
    await captureManager.CapturePhotoToStorageFileAsync(imgFormat, file);


    var image = Image.FromFile(file.Path);
    pictureBox1.Image = image;
}
```
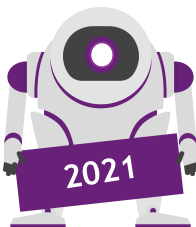
2021

Example 9 – Source Generators

- Autogenerate source-code during compilation step

- Based on Roslyn Analyzers

- Generated source-code is not part of your project, but is added to the compiled binary

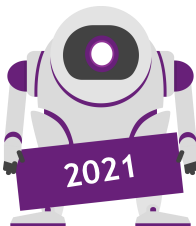- Source generators can be redistributed using NuGet packages

2021

Example 9 – Source Generators

```csharp
[Generator]
public class Generator : ISourceGenerator
{
    public void Initialize(GeneratorInitializationContext context)
    {
    }


    public void Execute(GeneratorExecutionContext context)
    {
    }
}
```
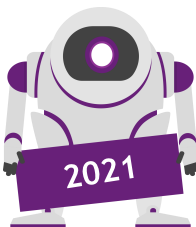
2021

Example 10 – Project Tye

- Experimental developer tool

- Better support for developing and testing distributed applications

- Run multiple services with a single command

- Service-address discovery

- Custom single-file configuration

- Support for .NET Core 3.1 onwards

2021

Example 10 – Project Tye

```yaml
name: projecttye
services:
- name: projecttye-backend
  project: 10-ProjectTye.Backend/10-ProjectTye.Backend.csproj
- name: projecttye-worker
  project: 10-ProjectTye.Worker/10-ProjectTye.Worker.csproj
- name: redis
  image: redis
  bindings:
  - port: 6379
    connectionString: "${host}:${port}"
- name: redis-cli
  image: redis
  args: "redis-cli -h redis MONITOR"
```
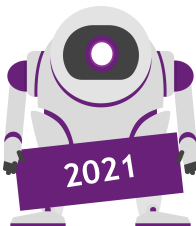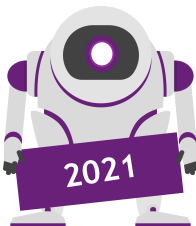
2021

SPONSORS

More information:
info@netcoreconf.com
@Netcoreconf

Visit on:
netcoreconf.com

Thank´s