C# 10

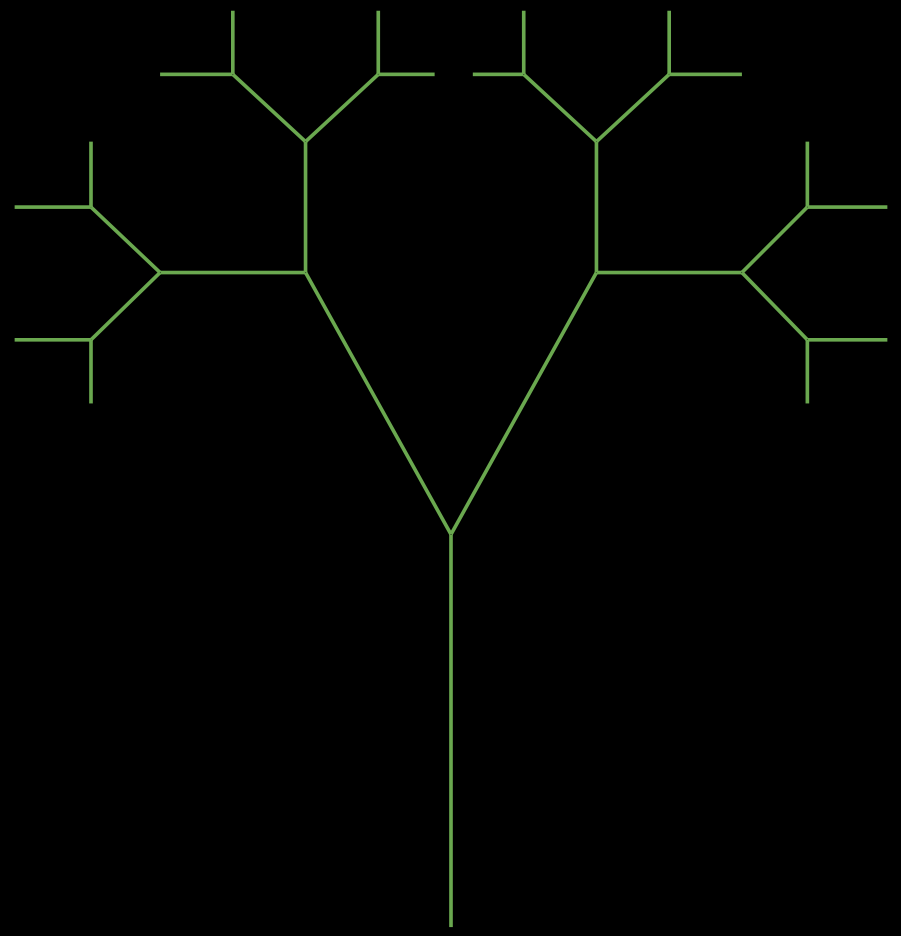
Rainer Stropek | @rstropek

C# 10 Status

- C# 10 has become available with .NET 6
- For C# >= 9, you need .NET >= 5 🔗
  - With .NET Core 3.x, you are limited to C# <= 8
- Interesting reads
-
  - C# Language Proposals
  - Language Design Meetings protocols

https://bit.ly/bc-cs10

Fractal Tree



Rules of the Game

- I am allowed to copy code (snippets)
  - Because of limited time
- Over-engineering is ok to demonstrate C# features
- Focus on language features, less class library
- We will not cover every minor detail
  - See slides and Microsoft docs for details
- Learn some things which are not directly related to C# news

record structs

Record structs proposal on GitHub, sample code on GitHub

Remember: *records*

```
1  using System;
2
3  var p = new Person("Foo", "Bar", 42);
4  // The following line does not work because records are immutable
5  // p.LastName = "Baz";
6  Console.WriteLine(p.FirstName);
7
8  var b = new Product("Bike", "Mountainbike", 499m);
9  Console.WriteLine(b.Name);
10
11 // Usual syntax, results in record classes (=reference types)
12 record Person(string FirstName, string LastName, int Age);
13
14 // New syntax "record class"
15 record class Product(string Category, string Name, decimal Price);
```

New: Explicitly mention turning record into class

https://sharplab.io/#gist:748161e42d7a0b840f1a86c03f2715ce

Record Structs are quite similar to Record Classes,
but there are differences, too

***IEquatable*, *ToString*, *Equals*, *GetHashCode*, you can add methods, etc.**

New: *record struct*

```
1  using System;
2
3  var v1 = new Vector2d(1d, 2d);
4  v1.X = 3d; // This works because get and set are generated by default
5  Console.WriteLine(v1.X);
6  Console.WriteLine(v1); // record structs implement ToString
7
8  var v2 = v1 with { X = 4d }; // We can use the with keyword
9  Console.WriteLine(v2.X);
10
11 Span<Vector2d> vectors = stackalloc Vector2d[]
12 {
13     new Vector2d(1d, 2d),
14     new Vector2d(3d, 4d),
15 };
16
17 //  New struct record (=value type)
18 record struct Vector2d(double X, double Y)
19 {
20     public static Vector2d operator +(Vector2d first, Vector2d second) =>
21         new Vector2d(first.X + second.X, first.Y + second.Y);
22 }
```

https://sharplab.io/#gist:b5000e681e1c7b495fef3d96ad319502

More About *record struct*

```
1  using System;
2  using System.Text;
3  using System.Text.Json;
4
5  var p1 = new Point(1d, 2d);
6  // p1.X = 3d; // This does not work because readonly records are immutable
7  Console.WriteLine(p1);
8
9  var (x, y) = p1; // Deconstruction works similar to record classes.
10
11 var p2 = p1 with { X = 4d }; // We can use the with keyword
12
13 // Readonly leads to an immutable struct
14 // Note that you can use the property: syntax to apply attributes
15 // or you can apply attributes to manually declared properties.
16 readonly record struct Point(double X, [property: JsonPropertyName("y")] double Y)
17 {
18     // It is possible to manually declare property.
19     [JsonPropertyName("x")]
20     public double X { get; init; } = X;
21
22     // Although PrintMembers is private, we can add a custom implementation.
23     // C# considers the members as "matching" if signature matches.
24     private bool PrintMembers(StringBuilder sb)
25     {
26         sb.Append($"X/Y = {X}/{Y}");
27         return true;
28     }
29 }
```

https://sharplab.io/#gist:7423c2a3753cba13c0be181216b9249b

More About *record struct*

```
1 record struct Vector3d(double X, double Y, double Z)
2 {
3     // We can turn properties into fields
4     // (works for record classes, too)
5     public double X = X;
6     public double Y = Y;
7     public double Z = Z;
8 }
```

Sealed Override *ToString*

General enhancement for records, not just record structs

```
1  using System;
2
3  var o = new TypeA("FooBar", 42);
4  Console.WriteLine(o); // Prints "FooBar" because of sealed override
5
6  public abstract record BaseRecord(string Name)
7  {
8      // The following line has not been possible before as
9      // BaseRecord is not sealed. Now, it is allowed.
10     public sealed override string ToString() => Name;
11 }
12
13 public sealed record TypeA(string Name, int Parameter) : BaseRecord(Name);
14 public sealed record TypeB(string Name, double Parameter) : BaseRecord(Name);
```

https://sharplab.io/#gist:076b6d63d6fd49eff33d1cd2f377006e

Global Using

Global Using proposal on GitHub

Global Using Directive

- `global using System.Text.Json;`
  - Works also with *using static*
- Motivation
  - Add a single file to your project with *using* directives that you frequently need
  - Makes *using* lists smaller in other files
  - Similar to Blazor's *_Imports.razor* file

Global Using

Imports.cs

```
1 global using System;
2 global using static System.Console;
3 global using System.Linq;
4 global using System.Text;
```

Program.cs

```
1  // Note: no `using System` necessary
2  var dates = new DateOnly[] {
3      new(2021, 1, 1),
4      new(2022, 1, 1)
5  };
6
7  // Note: no `using System.Text` necessary
8  var builder = new StringBuilder();
9
10 // Note: no `using System.Linq` necessary
11 var datesString = dates.Aggregate(
12     new StringBuilder(),
13     (sb, d) => sb.AppendLine(d.ToString("o")),
14     sb => sb.ToString());
15
16 // Note: no `Console` necessary
17 WriteLine(datesString);
```

Sample code on GitHub

Implicit *global using* directives

- To reduce the amount of *using* directives
- Enabled by default for .NET >= 6.0
  - Enabled in .csproj 🔗
  - You can disable it if you want 🔗
- List of namespaces see 🔗

File-scoped Namespaces

File-scoped Namespaces proposal on GitHub

File-scoped Namespaces

- Motivation
  - 99.99% of C# files contain a single namespace directive per file (source)
  - Why do we need the indentation? Code could be simplified.
  - Limitation: Single file-scoped namespace per file (source)

```
1  using System;
2
3  namespace MyApp
4  {
5    static class MyUtility
6    // Will compile into MyApp.MyUtility
7    {
8      public static String Greeting
9        => "Hi!";
10   }
11 }
```

```
1  using System;
2  namespace MyApp;
3
4  static class MyUtility
5  // Will compile into MyApp.MyUtility
6  {
7    public static String Greeting
8      => "Hi!";
9  }
```

https://sharplab.io/#gist:725b8fe4a643606ec57c2a841eac4da7

Convert to File-Scoped NS

Note: Auto-converting in latest VS2022

```
1   using System;
2
3   namespace Crm
4   {
        0 references
5       public class Person
6       {
            0 references
7           public static void SayHello()
8           {
9               Console.WriteLine("Hello");
10          }
11      }
12  }
13  }
```

Definite Assignment Improvements

Definite assignment improvements proposal on GitHub

Definite Assignment Improvements

```
1  #nullable enable
2  using static System.Console;
3
4  AnimalFactory? factory = new();
5
6  // Everything is fine in this case
7  if (factory != null && factory.TryGetAnimal(out Animal animal) && animal is Cat c)
8  {
9      WriteLine(c.Purr());
10 }
11
12 class AnimalFactory
13 {
14     public bool TryGetAnimal(out Animal animal)
15     {
16         animal = new Cat();
17         return true;
18     }
19 }
20
21 abstract class Animal { }
22 class Dog : Animal { public string Bark() => "Wuff"; }
23 class Cat : Animal { public string Purr() => "purrrrr"; }
```

https://sharplab.io/#gist:f45e55be16239b26cfdc81c49ea02167

Definite Assignment Improvements

```
1  // The following cases did not work before .NET 6
2
3  if (factory?.TryGetAnimal(out Animal animal2) == true && animal2 is Cat c2)
4  {
5      WriteLine(c2.Purr());
6  }
7
8  if (factory?.TryGetAnimal(out Animal animal3) is true && animal3 is Cat c3)
9  {
10     WriteLine(c3.Purr());
11 }
12
13 if ((factory?.TryGetAnimal(out Animal animal4) ?? false) && animal4 is Cat c4)
14 {
15     WriteLine(c4.Purr());
16 }
17
18 if ((factory != null ? factory.TryGetAnimal(out Animal animal5) : false)
19     && animal5 is Cat c5)
20 {
21     WriteLine(c5.Purr());
22 }
```

https://sharplab.io/#gist:f45e55be16239b26cfdc81c49ea02167

Static Abstract Members in Interfaces

Static abstract members proposal on GitHub

Motivation

- Create abstractions for static members in classes and structs
- Particularly important for static operators

Static Abstract Interface Members

```
1  using System;
2
3  ReadOnlySpan<Vector2d> vectors = stackalloc Vector2d[] { new(1d, 1d), new(2d, 2d), };
4  Console.WriteLine(AddAll(vectors));
5
6  static T AddAll<T>(ReadOnlySpan<T> addables) where T: IAddable<T>
7  {
8      var result = T.Zero;
9      foreach (var a in addables) result += a;
10     return result;
11 }
12
13 interface IAddable<T> where T : IAddable<T>
14 {
15     static abstract T Zero { get; }
16     static abstract T operator +(T t1, T t2);
17 }
18
19 record struct Vector2d(double X, double Y) : IAddable<Vector2d>
20 {
21     public static Vector2d operator +(Vector2d first, Vector2d second)
22         => new(first.X + second.X, first.Y + second.Y);
23     public static Vector2d Zero => new(0d, 0d);
24 }
```

https://sharplab.io/#gist:4148ab8042328489d93af3fdcec11441

Generic Math in C#

- Goal: Use operators on generic types (e.g. add two custom vector types)
  - See e.g. *INumber* 🔗
- Preview feature 🔗
  - You have to enable it
  - Add *<EnablePreviewFeatures>True </EnablePreviewFeatures>* in .csproj
  - Reference *System.Runtime.Experimental* NuGet

Read more

Operator Interface Name	Summary
IParseable	`Parse(string, IFormatProvider)`
ISpanParseable	`Parse(ReadOnlySpan<char>, IFormatProvider)`
IAdditionOperators	`x + y`
IBitwiseOperators	`x & y`, `x | y`, `x ^ y`, and `~x`
IComparisonOperators	`x < y`, `x > y`, `x <= y`, and `x >= y`
IDecrementOperators	`--x` and `x--`
IDivisionOperators	`x / y`
IEqualityOperators	`x == y` and `x != y`
IIncrementOperators	`++x` and `x++`
IModulusOperators	`x % y`
IMultiplyOperators	`x * y`
IShiftOperators	`x << y` and `x >> y`
ISubtractionOperators	`x - y`
IUnaryNegationOperators	`-x`
IUnaryPlusOperators	`+x`
IAdditiveIdentity	`(x + T.AdditiveIdentity) == x`
IMinMaxValue	`T.MinValue` and `T.MaxValue`
IMultiplicativeIdentity	`(x * T.MultiplicativeIdentity) == x`
IBinaryFloatingPoint	Members common to binary floating-point types
IBinaryInteger	Members common to binary integer types
IBinaryNumber	Members common to binary number types
IFloatingPoint	Members common to floating-point types
INumber	Members common to number types
ISignedNumber	Members common to signed number types
IUnsignedNumber	Members common to unsigned number types

```
1  public record struct Vector2d<T>(T X, T Y)
2      : IAdditionOperators<Vector2d<T>, Vector2d<T>, Vector2d<T>>,
3        IAdditionOperators<Vector2d<T>, T, Vector2d<T>>
4      where T : INumber<T>
5  {
6      public static Vector2d<T> operator +(Vector2d<T> left, Vector2d<T> right)
7          => new(left.X + right.X, left.Y + right.Y);
8
9      public static Vector2d<T> operator +(Vector2d<T> left, T delta)
10         => new(left.X + delta, left.Y + delta);
11 }
```

```
1  var v1 = new Vector2d<int>(1, 1);
2  var v2 = v1 + new Vector2d<int>(2, 2);
3  Assert.Equal(new Vector2d<int>(3, 3), v2);
4  ...
5
6  ...
7  var v1 = new Vector2d<int>(1, 1);
8  var v2 = v1 + 2;
9  Assert.Equal(new Vector2d<int>(3, 3), v2);
10 ...
```

```
1  ...
2  var vs = new[] { new Vector2d<int>(1, 1),
3                   new Vector2d<int>(2, 2) };
4  var sum = new Vector2d<int>(0, 0);
5  foreach (var v in vs)
6  {
7      sum += v;
8  }
9
10 Assert.Equal(new Vector2d<int>(3, 3), sum);
11 ...
```

λ Improvements

Lambda improvements proposal on GitHub

Lambda Improvements

```
1  using System;
2
3  var app = new EndpointConventionBuilder();
4
5  // Traditional way of defining a function with an attribute
6  [HttpGet("/")] int GetAnswer() => 42;
7  app.MapAction((Func<int>)GetAnswer);
8
9  // Now, we can remove the type cast:
10 app.MapAction(GetAnswer);
11
12 // We can even add attributes directly to lambdas:
13 app.MapAction([HttpGet("/")] () => 42);
```

https://sharplab.io/#gist:7b99f69d44d4ed4e9188f33ebd616a8a

Lambda Improvements

```
1  using System;
2
3  // In the past, we had to use explicit type for lambdas:
4  Func<int> f = () => 42;
5
6  // Lambdas will have a "natural type" that is compatible with var:
7  var f2 = () => 42;
8
9  // We will be able to call lambdas directly:
10 Console.WriteLine(((() => 42)()));
```

Enhancements related to Validations

Method parameter names in *nameof* proposal on GitHub

Caller Argument Expression proposal on GitHub

Constant Interpolated Strings proposal on GitHub

Simplified Parameter Null Validation proposal on GitHub

Declarations and Deconstruction proposal on GitHub

Parameter names in *nameof*

(was planned for C# 10, has been moved to *C# vNext*)

```
1 using System;
2 using System.Diagnostics.CodeAnalysis;
3
4 public class Path
5 {
6     [return: NotNullIfNotNull(nameof(path))]
7     public static string? GetFileName(string? path) { /* ... */ }
8 }
```

Constant Interpolated Strings

```
1  using System;
2
3  const string s1 = $"abc";
4  const string s2 = $"{s1}edf";
5  Console.WriteLine(s2);
6
7  DoSomething_Old(42);
8  DoSomething_VeryOld(42);
9
10 [Obsolete($"Use {nameof(DoSomething_New)} instead")]
11 void DoSomething_Old(int x) { }
12 void DoSomething_VeryOld(int x)
13 {
14     throw new InvalidOperationException(
15       $"{nameof(DoSomething_VeryOld)} is no longer supported");
16 }
17 void DoSomething_New(int x) { }
```

https://sharplab.io/#gist:06cb291efaf510a1641627901e2300cc

Caller Argument Expressions

```
1  #nullable enable
2
3  using System;
4  using System.Runtime.CompilerServices;
5
6  var x = 5;
7  Verify.Lower(x * 2, Convert.ToInt32(Math.Floor(Math.PI)));
8
9  public static class Verify
10 {
11     public static void Lower(int argument, int maxValue,
12         [CallerArgumentExpression("argument")] string? argumentExpression = null,
13         [CallerArgumentExpression("maxValue")] string? maxValueExpression = null)
14     {
15         if (argument > maxValue)
16         {
17             throw new ArgumentOutOfRangeException(nameof(argument),
18                 $"{argumentExpression} must be lower or equal {maxValueExpression}");
19         }
20     }
21 }
```

```
Exception
System.ArgumentOutOfRangeException: x * 2 must be lower or equal Convert.ToInt32(Math.Floor(Math.PI))
(Parameter 'argument')
   at Verify.Lower(Int32 argument, Int32 maxValue, String argumentExpression, String maxValueExpression)
   at <Program>$.<Main>$(String[] args)
```

https://sharplab.io/#gist:e883d7590ad7f2cc5e7c679fdabe51c4

Simplified Null Validation

(was planned for C# 10, has been moved to *C# vNext*)

```
// Before
void Insert(string s) {
  if (s is null)
    throw new ArgumentNullException(nameof(s));

  ...
}

// After
void Insert(string s!!) {
  ...
}
```

Declarations and Deconstruction

```
1  #nullable enable
2
3  using System;
4
5  Err err;
6
7  // Note that we can now mix declaration and tuple deconstruction
8  (var ret1, err) = GetAnswer();
9  if (err == null) Console.WriteLine(ret1);
10
11 // Go-like error handling anybody?
12 (var ret2, err) = GetAnswer_Error();
13 if (err != null) Console.WriteLine(err);
14
15 (int?, Err?) GetAnswer() => (42, null);
16 (int?, Err?) GetAnswer_Error() => (null, new());
17
18 class Err { public string Message => "Error"; }
```

https://sharplab.io/#gist:d090ecba8dac70006f184377ff1a89a3

Pattern Matching Enhancements

Extended Property Patterns proposal on GitHub

Extended Property Patterns

```
1  using System;
2  using System.Collections.Generic;
3
4  var heroes = new List<IHero>
5  {
6      new Hero("Homelander", new(true, true)),
7      new Hero("Stormfront", new(true, false)),
8  };
9
10 // Old: if (heroes[0] is Hero { Flying: { CanFlyInSpace: true } })
11 if (heroes[0] is Hero { Flying.CanFlyInSpace: true })
12 {
13     Console.WriteLine("Hero can fly on earth");
14 }
15
16 class Flying
17 {
18     public bool CanFlyOnEarth { get; set; }
19     public bool CanFlyInSpace { get; set; }
20     public Flying(bool canFlyOnEarth, bool canFlyInSpace)
21         => (CanFlyOnEarth, CanFlyInSpace) = (canFlyOnEarth, canFlyInSpace);
22 }
23
24 interface IHero { }
25 class Hero : IHero
26 {
27     public string Name;
28     public Flying Flying;
29     public Hero(string name, Flying flying)
30         => (Name, Flying) = (name, flying);
31 }
```

https://sharplab.io/#gist:d65641de6e913b8de6592a756709b244

String Interpolation Enhancements

Read more

Demo Time!

Kahoot

C# 10 🤘

Rainer Stropek | @rstropek