

What's New in C# 10

Jason Bock

Personal Info

- <http://www.jasonbock.net>
- <https://www.twitter.com/jasonbock>
- <https://www.github.com/jasonbock>
- <https://www.youtube.com/c/JasonBock>
- jason.r.bock@outlook.com

Downloads

<https://github.com/JasonBock/WhatsNewInCSharp10>

<https://github.com/JasonBock/Presentations>

Overview

- Language Evolution
- C# 10 Features
- Future Directions

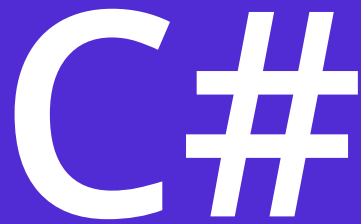
Remember...

<https://github.com/JasonBock/WhatsNewInCSharp10>

<https://github.com/JasonBock/Presentations>

Language Evolution

What's New in C# 10

The C# logo is displayed in white on a solid blue rectangular background. It consists of a large, bold, sans-serif capital letter 'C' followed by a bold, sans-serif hash symbol '#'. The two characters are closely spaced and share a common baseline.

Est. 2002

It's hard to believe that C# has been around since 2002. Along the way it's picked up a fair amount of features.

Version 1

Classes	Structs	Interfaces	Events	Properties
Delegates	Expressions	Statements	Attributes	Literals

Version 1 had some “standard” language features that one would arguably expect from an OOP language.

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md#c-10-visual-studionet>

Version 2

Generics	Partial types	Anonymous methods	Iterators	Nullable types
Getter/setter separate accessibility	Method group conversions (delegates)	Co- and Contra-variance for delegates and interfaces	Static classes	Delegate inference

Version 2's biggest addition was generics. That alone made coding easier to write, especially when it came to collections.

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md#c-2-vs-2005>

Version 3

Implicitly
typed local
variables

Object and
collection
initializers

Auto-
Implemented
properties

Anonymous
types

Extension
methods

Query
expressions

Lambda
expression

Expression
trees

Partial
methods

LINQ was the biggest addition for C#3. With it came a whole slew of features to support it.

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md#c-3-vs-2008>

Version 4

Dynamic
binding

Named and
optional
arguments

Generic co-
and
contravariance

Embedded
interop types
("NoPIA")

Version 4 added some “interesting” features, with dynamic opening the door to C# being a little less type-safe.

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md#c-4-vs-2010>

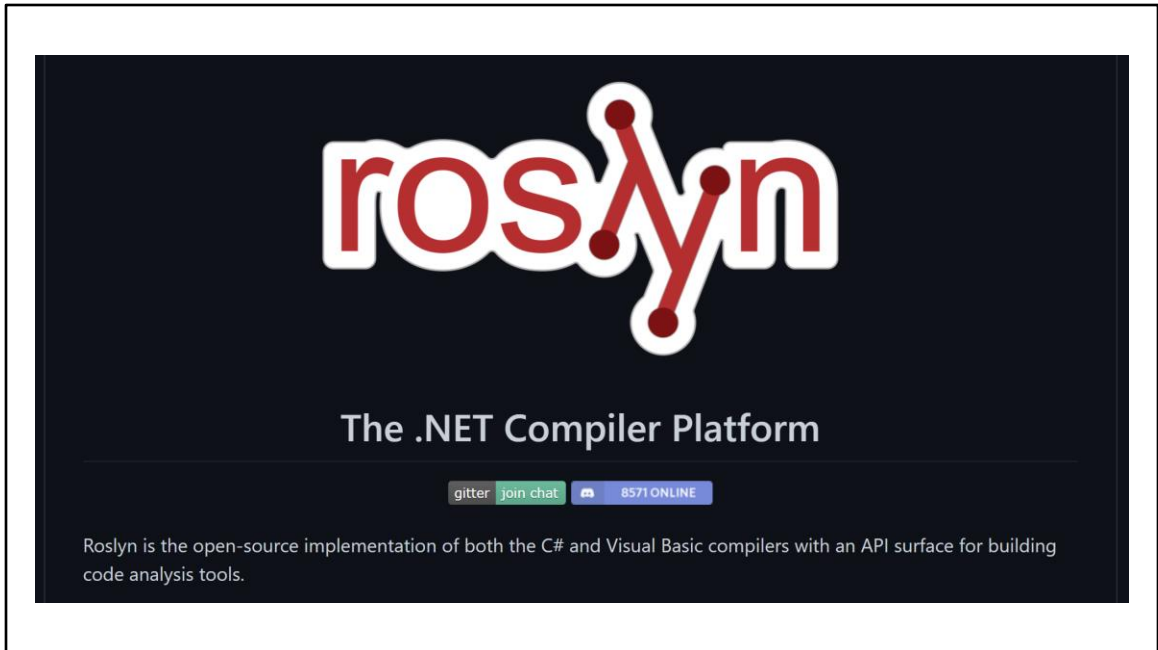
Version 5

Asynchronous
methods

Caller info
attributes

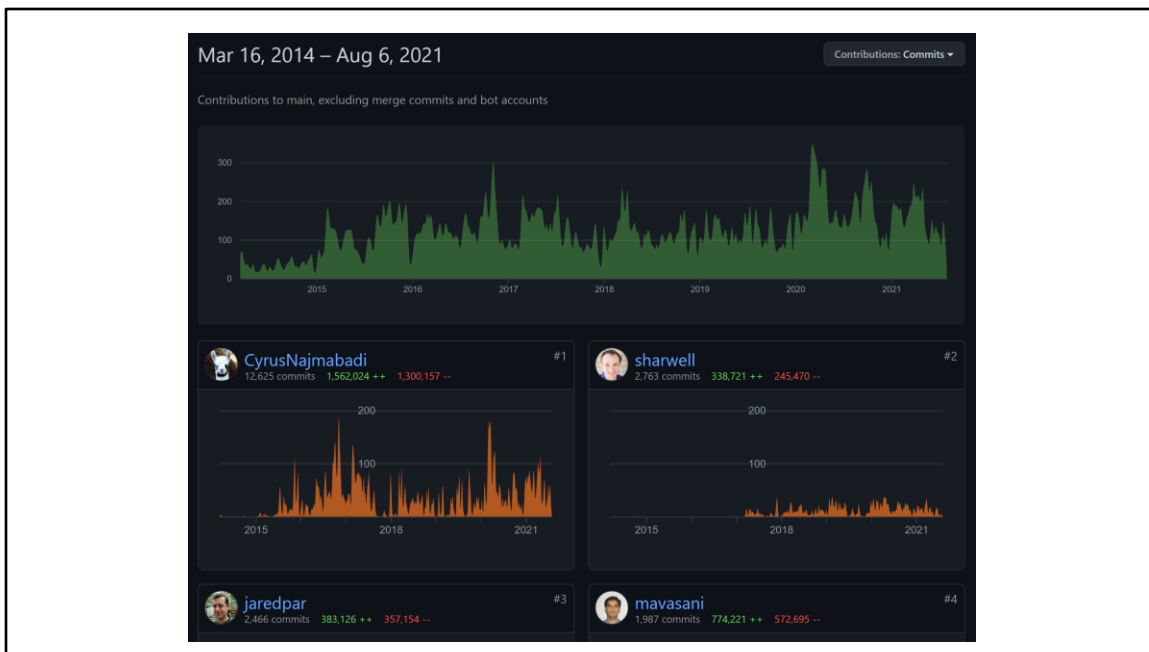
Version 5's biggest (and maybe only) feature is async/await, making asynchronous/concurrent programming easier.

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md#c-5-vs-2012>



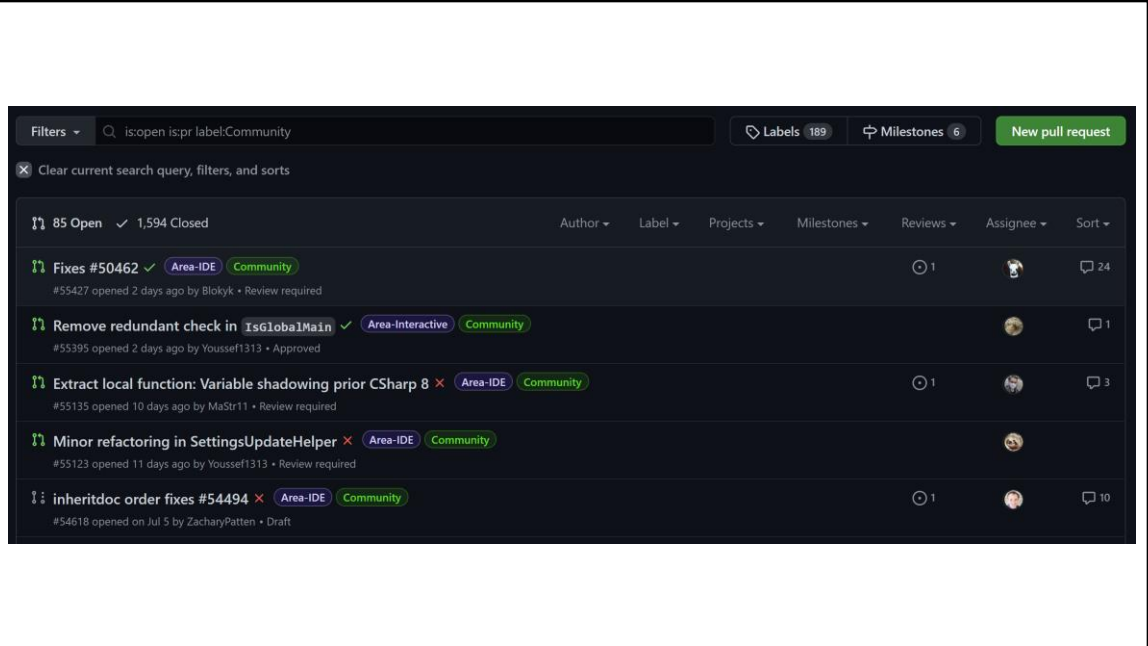
The language space is open source. You can download the source code to the compilers and hack away (and submit pull requests if you'd like). You can watch as the language teams discuss current and potentially future changes. This is completely unlike the .NET of 2002.

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



You can see who has contributed to Roslyn.

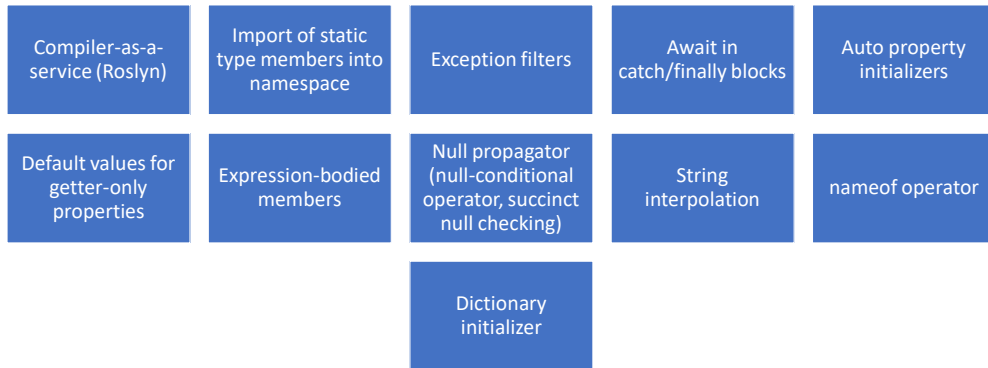
<https://github.com/dotnet/roslyn/graphs/contributors>



There's a "Community" tag so you can see which PRs have been tagged with that.

<https://github.com/dotnet/roslyn/pulls?q=is%3Aopen+is%3Apr+label%3ACommunity>

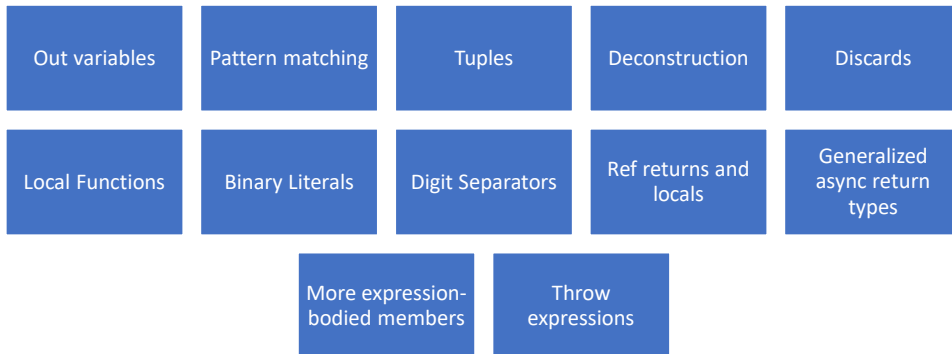
Version 6



With Version 6, there's a lot of small, yet useful, features. And it's interesting to note that a fair amount of these (like string interpolation) went through a lot of changes before it reached its final destination. Some features, like primary constructors, have been dropped for now (though they may come back in a future version). All this work was driven a lot by the community and what really worked best.

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md#c-6-vs-2015>

Version 7



C#7 has continued the trend from C#6, with lots of small, yet useful features.

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md#c-70-visual-studio-2017>

Version 7.1

Async main

Default
expressions

Reference
assemblies

Inferred tuple
element
names

Pattern-
matching with
generics

But C#7 has bucked a trend by introducing point releases. C#7.1 was released with the VS2017 15.3 release, and we'll cover those as well in this talk

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md#c-71-visual-studio-2017-version-153>

<https://github.com/dotnet/roslyn/blob/main/docs/Language%20Feature%20Status.md#c-71>

Version 7.2

Ref readonly

Interior
pointer/Span/ref
struct

Non-trailing
named
arguments

private protected

Conditional ref
operator

Digit separator
after base
specifier

7.2 continued the point trend.

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md#c-72---visual-studio-2017-version-155>

<https://github.com/dotnet/roslyn/blob/main/docs/Language%20Feature%20Status.md#c-72>

<https://github.com/dotnet/roslyn/blob/main/docs/Language%20Feature%20Status.md#c-72-fixes>

Version 7.3

Enum, delegate,
and unmanaged
constraints

Ref local re-
assignment

Stackalloc
initializers

Indexing movable
fixed buffers

Custom fixed
statement

Improved overload
candidates

Expression
variables in
initializers and
queries

Tuple comparison

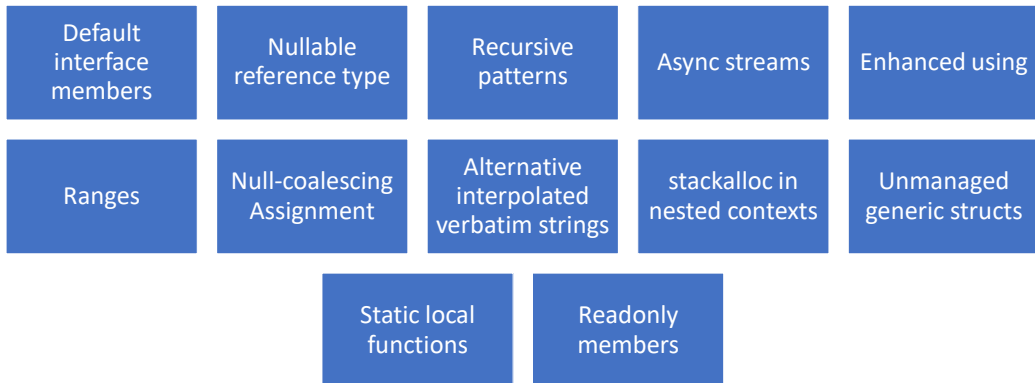
Attributes on
backing fields

So did 7.3

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md#c-73---visual-studio-2017-version-157>

<https://github.com/dotnet/roslyn/blob/main/docs/Language%20Feature%20Status.md#c-73>

Version 8



8.0 was released with .NET Core 3.0

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md#c-80---net-core-30-and-visual-studio-2019-version-163>

<https://github.com/dotnet/roslyn/blob/main/docs/Language%20Feature%20Status.md#c-80>

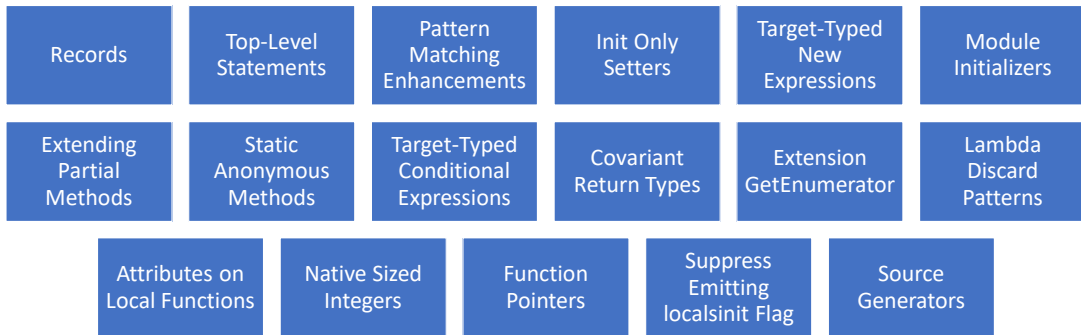
Many of the C# 8.0 language features have platform dependencies. Async streams, indexers and ranges all rely on new framework types that will be part of .NET Standard 2.1. As Immo describes in his post [Announcing .NET Standard 2.1](#), .NET Core 3.0 as well as Xamarin, Unity and Mono will all implement .NET Standard 2.1, but .NET Framework 4.8 will not. This means that the types required to use these features won't be available on .NET Framework 4.8. Likewise, default interface member implementations rely on new runtime enhancements, and we will not make those in the .NET Runtime 4.8 either.

For this reason, **using C# 8.0 is only supported on platforms that implement .NET Standard 2.1**. The need to keep the runtime stable has prevented us from implementing new language features in it for more than a decade. With the side-by-side and open-source nature of the modern runtimes, we feel that we can responsibly evolve them again, and do language design with that in mind. Scott explained in his [Update on .NET Core 3.0 and .NET Framework 4.8](#) that .NET Framework is going to see less innovation in the future, instead focusing on stability and reliability. Given that, we think it is better for it to miss out on some language features than for nobody to get them.

Note that C#8 won't fully work in .NET Framework 4.8

<https://devblogs.microsoft.com/dotnet/building-c-8-0/#platform-dependencies>
https://unsplash.com/photos/4Zaq5xY5M_c

Version 9



Note that C# 9 (and F# 5) target .NET 5.

<https://github.com/dotnet/csharpplang/blob/master/Language-Version-History.md#c-90---net-5-and-visual-studio-2019-version-168>

<https://github.com/dotnet/roslyn/blob/main/docs/Language%20Feature%20Status.md#c-9>

Version 10 (preview)

Records Structs	Global Using Directive	Improved Definite Assignment	Constant Interpolated Strings	Extended Property Patterns	Sealed Record ToString
Source Generator V2 APIs	Mix Declarations and Variables in Deconstruction	Async Method Builder Override	Enhanced #line Directive	Lambda Improvements	Static Abstract Members In Interfaces C# 10 Preview
	Interpolated String Improvements	File-Scoped Namespace	Parameterless Struct Constructors	Caller Expression Attribute	

And now we have 10.0, which will target .NET 6

<https://github.com/dotnet/csharpplang/blob/main/Language-Version-History.md#c-100---net-6-and-visual-studio-2022-version-170>

<https://github.com/dotnet/roslyn/blob/main/docs/Language%20Feature%20Status.md#c-100>



Again, this is in preview and the final version will be released Nov. 2021. Things may change from now to then.

<https://www.pexels.com/photo/photo-of-pathway-surrounded-by-fir-trees-1578750/>

Demo: C# 10 Features

What's New in C# 10

Version 10 (preview)

Records Structs	Global Using Directive	Improved Definite Assignment	Constant Interpolated Strings	Extended Property Patterns	Sealed Record ToString
Source Generator V2 APIs	Mix Declarations and Variables in Deconstruction	Async Method Builder Override	Enhanced #line Directive	Lambda Improvements	Static Abstract Members In Interfaces C# 10 Preview
	Interpolated String Improvements	File-Scoped Namespace	Parameterless Struct Constructors	Caller Expression Attribute	

To recap, here are the features in C# 10

Future Directions

What's New in C# 10



So, where will C# go from here? As with any crystal ball gazing, sometimes the best we can do is guess. But with C# being OSS, it's easier to see the roadmap, so let's talk about some of the features that may show up in the future.

<https://unsplash.com/photos/GY38n9WKjQI>

Version “Next”

nameof(parameter)

Relax ordering of ref
and partial modifiers

Parameter null-
checking

Generic attributes

Default in
deconstruction

List patterns

Raw string literals

These may get in some future version of C#, or not, we don't know yet.

<https://github.com/dotnet/roslyn/blob/master/docs/Language%20Feature%20Status.md#c-next>

C# Language Design

chat on [gitter](#) 8571 ONLINE

Welcome to the official repo for C# language design. This is where new C# language features are developed, adopted and specified.

C# is designed by the C# Language Design Team (LDT) in close coordination with the [Roslyn](#) project, which implements the language.

You can find:

- Active C# language feature proposals in the [proposals folder](#)
- Notes from C# language design meetings in the [meetings folder](#)
- Full C# 6 language specification (draft) in the [spec folder](#)
- Summary of the [language version history here](#).

This document describes the language design process, and how they are categorized

<https://github.com/dotnet/csharplang/blob/master/README.md>

Working Set

No due date 2% complete

These proposals will be or are being designed by the Language Design Team during the current design timeframe. Not all the proposals in this bucket will actually make it into the language for the next version of C#, but they will get some design time from the team.

49 Open 1 Closed

<div> <div></div> <div>Champion "Type Classes (aka Concepts, Structural Generic Constraints)"</div> <div>#110 opened on Feb 14, 2017 by galfert 5 tasks</div> </div> <div> <div>Long lead</div> <div>Proposal champion</div> </div> <div> <div></div> <div>188</div> </div>
<div> <div></div> <div>Proposal: "Closed" type hierarchies</div> <div>#485 opened on Apr 21, 2017 by galfert</div> </div> <div> <div>Feature Request</div> <div>Proposal</div> </div> <div> <div></div> <div>21</div> </div>
<div> <div></div> <div>generic constraint: where T : ref struct</div> <div>#1148 opened on Nov 25, 2017 by lucasmeljer</div> </div> <div> <div>Feature Request</div> <div>Proposal champion</div> </div> <div> <div></div> <div>37</div> </div>
<div> <div></div> <div>C# Feature Request: Allow value tuple deconstruction with default keyword</div> <div>#1358 opened on Mar 6, 2018 by KnorrThieus</div> </div> <div> <div>Proposal champion</div> </div> <div> <div></div> <div>33</div> </div>
<div> <div></div> <div>Five ideas for improving working with fixed buffers</div> </div> <div> <div>Feature Request</div> <div>Proposal champion</div> </div> <div> <div></div> <div>22</div> </div>

This is another resource to look at language ideas and planning. The “Type Classes” one I’m hoping is eventually done.

<https://github.com/dotnet/csharplang/milestone/19>

Developing a Language Feature

Adding a new feature to C# or VB is a very serious undertaking that often takes several iterations to complete for even the (seemingly) simplest of features. This is due to both the inherent complexity of changing languages and the need to consider the effects of new features in all layers of the Roslyn codebase: IDE, debugging, scripting, etc. As such, language work occurs in a separate branch until the feature reaches a point when we are ready to merge it into the main compiler.

This page discusses the process by which language feature *implementations* are considered, prototyped, and fully accepted into the language. This process is intended to be used by the compiler team and community.

Process

1. **Feature specification filed:** The speclet should be filed as a GitHub issue and contain:

- * A description of the feature (including any syntax changes involved)
- * Discussions about impacted areas, such as overload resolution and type inference. Think through the major areas of the language
- * Proposed changes to the API surface area.

A feature speclet is different from a language design discussion. Discussions are very open-ended and often for features that simply won't

If you have an idea, submit a feature request and see where it goes. Just follow the directions....also, keep in mind that if your idea will touch overloading in any way, the review process gets really complicated.

<https://github.com/dotnet/roslyn/blob/master/docs/contributing/Developing%20a%20Language%20Feature.md>



YOU can influence the future direction if you want! The “more expressions-bodied members” was proposed and implemented by the community, so you can get involved!

<https://unsplash.com/photos/7iatBuqFvY0>

What's New in C# 10

Jason Bock

Remember...

- <https://github.com/JasonBock/WhatsNewInCSharp10>
- <https://github.com/JasonBock/Presentations>
- References in the notes on this slide

References

Features Added in C# Language Versions -

<https://github.com/dotnet/csharplang/blob/master/Language-Version-History.md>

Language Feature Status -

<https://github.com/dotnet/roslyn/blob/master/docs/Language%20Feature%20Status.md>

C# Guide - <https://docs.microsoft.com/en-us/dotnet/csharp/>

What's the strangest corner case you've seen in C# or .NET? -

<https://stackoverflow.com/questions/194484/whats-the-strangest-corner-case-youve-seen-in-c-sharp-or-net>

Eliminating Nulls in C# - <https://magenic.com/thinking/eliminating-nulls-in-c>

Language Version Planning - <https://github.com/dotnet/csharplang/projects/4>

Language Design Meetings -

<https://github.com/dotnet/csharplang/tree/master/meetings>

C#.next

(<https://github.com/dotnet/roslyn/blob/master/docs/Language%20Feature%20Status.md#c-next>)

- * Caller expression attribute
- * Generic attributes
- * Default in deconstruction

C#9

* [C# 9 - Improving performance using the SkipLocalsInit attribute](<https://www.meziantou.net/csharp-9-improve-performance-using-skiplocalsinit.htm>)

* JsonSrcGen + CoreRT = Pure Magic -

<https://trampster.blogspot.com/2020/09/jsonsrcgen-core-rt-pure-magic-in-my.html>

*

<https://github.com/dotnet/roslyn/blob/master/docs/Language%20Feature%20Status.md#c-9>

* <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/proposals/csharp-9.0/records>

* Reduce mental energy with C# 9 -

<https://daveabrock.com/2020/06/18/reduce-mental-energy-with-c-sharp>

* Source Generator

* Introducing C# Source Generators -

<https://devblogs.microsoft.com/dotnet/introducing-c-source-generators/>

* Source Generators Cookbook -

<https://github.com/dotnet/roslyn/blob/master/docs/features/source-generators.cookbook.md>

* Demos - <https://github.com/dotnet/roslyn-sdk/tree/master/samples/CSharp/SourceGenerators>

* INotifyPropertyChanged with C# 9.0 Source Generators - <https://jaylee.org/archive/2020/04/29/notify-property-changed-with-roslyn-generators.html>

* C# Source Generators: Less Boilerplate Code, More Productivity - <https://dontcodetired.com/blog/post/C-Source-Generators-Less-Boilerplate-Code-More-Productivity>

* Using C# Source Generators with Microsoft Feature Management Feature Flags - <https://dontcodetired.com/blog/post/Using-C-Source-Generators-with-Microsoft-Feature-Management-Feature-Flags>

* StackOnlyJsonParser - <https://github.com/TomaszRewak/C-sharp-stack-only-json-parser>

* First Look: C# Source Generators -

<https://daveabrock.com/2020/05/08/first-look-c-sharp-generators>

* Source Generators in

C#](<https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/Source-Generators-in-CSharp>)

- * Target-typed new
- * Relax ordering of ref and partial modifiers
- * Parameter null-checking
- * Skip locals init
- * Lambda discard parameters
- * Native ints
- * Attributes on local functions
- * Function pointers
- * Pattern matching improvements
- * Static lambdas
- * Records
- * Target-typed conditional
- * Covariant Returns
- * Extension GetEnumerator
- * Module initializers
- * Extending Partial
- * Top-level statements

C#8

- * Nullable reference types

*

<https://github.com/dotnet/roslyn/blob/master/docs/features/nullable-reference-types.md>

<https://github.com/dotnet/roslyn/blob/master/docs/features/nullable-metadata.md>

* <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/proposals/csharp-8.0/nullable-reference-types-specification>

* .NET Docs - <https://docs.microsoft.com/en-us/dotnet/csharp/nullable-references>

* Embracing nullable reference types - <https://devblogs.microsoft.com/dotnet/embracing-nullable-reference-types/>

* Reserved attributes contribute to the compiler's null state static analysis - <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/attributes/nullable-analysis>

* Ian Griffiths series - <https://endjin.com/blog/2020/04/dotnet-csharp-8-nullable-references-non-nullable-is-the-new-default>

- * C# 8 Interfaces

* Static Members - <https://jeremybytes.blogspot.com/2019/12/c-8-interfaces-static-members.html>

* Public, Private, and Protected Members - <https://jeremybytes.blogspot.com/2019/11/c-8-interfaces-public-private-and.html>

*** Dangerous Assumptions in Default Implementation -**
<https://jeremybytes.blogspot.com/2019/09/c-8-interfaces-dangerous-assumptions-in.html>

*** A Closer Look at C# 8 Interfaces -**
<https://jeremybytes.blogspot.com/2019/09/a-closer-look-at-c-8-interfaces.html>

*** Interfaces in C# 8 are a Bit of a Mess -**
<https://jeremybytes.blogspot.com/2019/09/interfaces-in-c-8-are-bit-of-mess.html>

Essential C# 8.0 - <https://www.codemag.com/article/1911112>

C#, Span and async - <https://blogs.endjin.com/2019/10/c-span-and-async/>

C# 8.0 and .NET Standard 2.0 - Doing Unsupported Things -
<https://stu.dev/csharp8-doing-unsupported-things/>

"deferring" in C#8 -

<https://twitter.com/reubenbond/status/1184135702851678208>

What's New in C#8 - <https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-8>

This is how you get rid of null reference exceptions forever -

<https://channel9.msdn.com/Shows/On-NET/This-is-how-you-get-rid-of-null-reference-exceptions-forever>

Alignment with C# 8 Switch Expressions -

<https://csharp.christiannagel.com/2019/09/17/switchexpressionalign/>

Async Streams – A Look at New Language Features in C# 8 -

<https://blog.jetbrains.com/dotnet/2019/09/16/async-streams-look-new-language-features-c-8/>

Eliminating Nulls in C# - <https://magenic.com/thinking/eliminating-nulls-in-c-dim>

Default Interface Members and Inheritance -

<https://daveaglick.com/posts/default-interface-members-and-inheritance>

Default Interface Members, What Are They Good For? -

<https://daveaglick.com/posts/default-interface-members-what-are-they-good-for>

Update libraries to use nullable reference types and communicate nullable rules to callers - <https://docs.microsoft.com/en-us/dotnet/csharp/nullable-attributes>

Try out Nullable Reference Types - <https://devblogs.microsoft.com/dotnet/try-out-nullable-reference-types/>

C# 8 Pattern Matching Enhancements -

<https://www.infoq.com/news/2019/06/CSharp-8-Pattern-Matching/>

Tutorial: Using pattern matching features to extend data types -

<https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/pattern-matching>

Default implementations in interfaces -

<https://devblogs.microsoft.com/dotnet/default-implementations-in-interfaces/>

C#8, .NET Framework and .NET Core Divergence -

<https://twitter.com/dotMorten/status/1105343335164596224?s=09>

Async Streams with C#8 -

<https://csharp.christiannagel.com/2019/03/20/asyncstreams/>

.NET Core 3.0 vs. .NET Framework -

<https://twitter.com/dotMorten/status/1105343335164596224?s=09>

Much Ado about Nothing: A C# play in two acts.

Part 1 - https://www.youtube.com/watch?v=GusJQNjj_2g

Part 2 - <https://www.youtube.com/watch?v=IVDYwA-BcwE>

Adapting Projects to Use C# 8 and Nullable Reference Types -

<https://www.infoq.com/articles/csharp-nullable-reference-case-study>

NullableAttribute and C#8 -

<https://codeblog.jonskeet.uk/2019/02/10/nullableattribute-and-c-8/>

Update on IAsyncDisposable and IAsyncEnumerator -

<https://www.infoq.com/news/2019/01/IAsyncDisposable-IAsyncEnumerator>

Do more with patterns in C# 8.0 -

<https://blogs.msdn.microsoft.com/dotnet/2019/01/24/do-more-with-patterns-in-c-8-0/>

Take C# 8.0 for a spin -

<https://blogs.msdn.microsoft.com/dotnet/2018/12/05/take-c-8-0-for-a-spin/>

Containing Null with C# 8 Nullable References -

<https://praeclarum.org/2018/12/17/nullable-reference-types.html>

C# 8: Pattern Matching Extended -

<https://csharp.christiannagel.com/2018/07/03/patternmatchingcs8/>

Writing operators - <https://github.com/akarnokd/async-enumerable-dotnet/wiki/Writing-operators>

Optimizing C# Struct Equality with IEquatable and ValueTuples -

<https://montemagno.com/optimizing-c-struct-equality-with-iequatable/>

C# 8 - Jon Skeet and Mads Torgersen -

<https://www.youtube.com/watch?v=gGUYUJmssYM>

The future of C# (Build 2018) -

<https://channel9.msdn.com/Events/Build/2018/BRK2155>

First steps with nullable reference types -

<https://codeblog.jonskeet.uk/2018/04/21/first-steps-with-nullable-reference-types/>

A Preview of C# 8 with Mads Torgersen -

<https://channel9.msdn.com/Blogs/Seth-Juarez/A-Preview-of-C-8-with-Mads-Torgersen>

Introducing Nullable Reference Types in C# -

<https://blogs.msdn.microsoft.com/dotnet/2017/11/15/nullable-reference-types-in-csharp/>

Herding Nulls and Other C# Stories From the Future -

<https://www.infoq.com/presentations/c-sharp-future>

Before C#8

How Microsoft rewrote its C# compiler in C# and made it open source -

<https://medium.com/microsoft-open-source-stories/how-microsoft-rewrote-its-c-compiler-in-c-and-made-it-open-source-4ebed5646f98>

Tuples, deconstruction, and dictionaries -

<https://twitter.com/davidpine7/status/1032706884569059328>

Interesting uses of tuple deconstruction -

<https://compiledexperience.com/blog/posts/abusing-tuples>

Performance traps of ref locals and ref returns in C# -

<https://blogs.msdn.microsoft.com/seteplia/2018/04/11/performance-traps-of-ref-locals-and-ref-returns-in-c/>

Memory Management, C# 7.2 and Span<T> -

<https://speakerdeck.com/slang25/memory-management-c-number-7-dot-2-and-span>

Using In Parameter Modifier - C# 7.2 - <https://codewala.net/2018/03/27/using-in-parameter-modifier-c-7-2/>

Essential .NET - C# 8.0 and Nullable Reference Types -

<https://msdn.microsoft.com/en-us/magazine/mt829270.aspx>

C# 7 Series, Part 6: Read-only structs -

<https://blogs.msdn.microsoft.com/mazhou/2017/11/21/c-7-series-part-6-read-only-structs/>

What's new in C# 7.2 - What's new in C# 7.2

What's new in C# 7.2 - <https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-7-2>

C# 7.2: Understanding Span -

<https://channel9.msdn.com/Events/Connect/2017/T125>

C# Language Internals - <https://www.youtube.com/watch?v=1lnwO63LhRI>

Dissecting the local functions in C# 7 -

<https://blogs.msdn.microsoft.com/seteplia/2017/10/03/dissecting-the-local-functions-in-c-7/>

C# 7.1 - Everything You Need To Know -

<https://www.danielcrabtree.com/blog/329/c-sharp-7-1-everything-you-need-to-know>

C# 7, 8 and Beyond: Language Features from Design to Release to IDE Support -

<https://www.infoq.com/presentations/csharp-7-8-language-features>

Perusing C# 7.1 - <http://davidpine.net/blog/csharp-seven-dot-one/>

Exploring C# 7 - <http://davidpine.net/blog/exploring-csharp-seven/>

Tuples and Generics - <https://magenic.com/thinking/tuples-and-generics-in-c7>

Tuples and Deconstruction in C#7 - <https://magenic.com/thinking/tuples-and-deconstruction-in-c7>

C#7 Series

Value Tuples:

<https://blogs.msdn.microsoft.com/mazhou/2017/05/26/c-7-series-part-1-value-tuples/>

Async Main:

<https://blogs.msdn.microsoft.com/mazhou/2017/05/30/c-7-series-part-2-async-main/>

Default Literals:

<https://blogs.msdn.microsoft.com/mazhou/2017/06/06/c-7-series-part-3-default-literals/>

Discards:

<https://blogs.msdn.microsoft.com/mazhou/2017/06/27/c-7-series-part-4-discards/>

Private Protected:

<https://blogs.msdn.microsoft.com/mazhou/2017/10/05/c-7-series-part-5-private-protected/>

Read-Only Structs:

<https://blogs.msdn.microsoft.com/mazhou/2017/11/21/c-7-series-part-6-read-only-structs/>

Ref Returns:

<https://blogs.msdn.microsoft.com/mazhou/2017/12/12/c-7-series-part-7-ref-returns/>

"in" Parameters:

<https://blogs.msdn.microsoft.com/mazhou/2018/01/08/c-7-series-part-8-in-parameters/>

Ref Structs:

<https://blogs.msdn.microsoft.com/mazhou/2018/03/02/c-7-series-part-9-ref-structs/>

Span<T> and universal memory management:

<https://blogs.msdn.microsoft.com/mazhou/2018/03/25/c-7-series-part-10-span-and-universal-memory-management/>

Expression – Bodied Members in C# 7.0 -

<http://dailydotnettips.com/2017/07/17/expression-bodied-members-in-c-7-0/>

Abolishing Switch-Case Statement and Pattern Matching in C# 7.0:

<https://rubikskode.net/2017/06/18/abolishing-switch-case-and-pattern-matching-in-c-7-0/>

C# 7 – Simplify and Improve Your Code in 2017:

<https://channel9.msdn.com/Events/DEVintersection/DEVintersection-2017-Orlando/DEV005>

C# 7.x and 8.0: Uncertainty and Awesomeness: <https://www.erikheemskerk.nl/c-sharp-7-2-and-8-0-uncertainty-awesomeness/>

C# 7.1 and Beyond: Polishing Usability - <https://www.erikheemskerk.nl/c-sharp-7-1-polishing-usability/>

C# 7.2 and 8.0 Roadmap - <https://www.infoq.com/news/2017/06/CSharp-7.2>

An Early Look at C# 7.1: Part 1 - <https://www.infoq.com/news/2017/06/CSharp-7.1-a>

An Early Look at C# 7.1: Part 2 - <https://www.infoq.com/news/2017/06/CSharp-7.1-b>

The Future of C# - <https://channel9.msdn.com/Events/Build/2017/B8104>

What's new in C# 7 - <https://docs.microsoft.com/en-us/dotnet/articles/csharp/whats-new/csharp-7>

Patterns and Practices in C# 7 - <https://www.infoq.com/articles/Patterns-Practices-CSharp-7>

New Features in C# 7.0 -

<https://blogs.msdn.microsoft.com/dotnet/2017/03/09/new-features-in-c-7-0/>

C#7 Tools (see other related articles from this author) -

<http://blog.somewhatabstract.com/2017/02/20/c7-tools/>

.NET Framework - What's New in C# 7.0 - <https://msdn.microsoft.com/en-us/magazine/mt790184.aspx>

Nested Functions: https://en.wikipedia.org/wiki/Nested_function (think function to delete files, needs recursion, local functions makes this elegant, but is it performant?)

C# 7 Work List of Features - <https://github.com/dotnet/roslyn/issues/2136>

What's New in C#7 -

<https://blogs.msdn.microsoft.com/dotnet/2016/08/24/whats-new-in-csharp-7-0/>

Language Feature Status -

<https://github.com/dotnet/roslyn/blob/master/docs/Language%20Feature%20Status.md>

Visual Studio "15" Preview 5 -

<https://www.visualstudio.com/news/releasenotes/vs15-relnotes#cshappvb>

Language Feature Status -

<https://github.com/dotnet/roslyn/blob/master/docs/Language%20Feature%20Status.md>