

Writing Modern C#

Jason Bock
Staff Software Engineer
Rocket Mortgage

Personal Info

- <http://jasonbock.net>
- <https://mstdn.social/@jasonbock>
- <https://github.com/jasonbock>
- <https://youtube.com/JasonBock>
- <https://www.linkedin.com/in/jasonrbock/>
- jason.r.bock@outlook.com

Downloads

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

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

Overview

- Language Evolution
- Using Modern C# Features
- Call to Action

Remember...

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

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

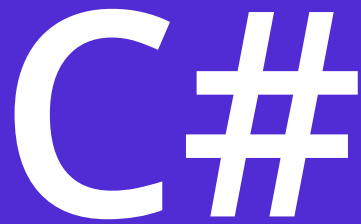
Language Evolution

Writing Modern C#

C# is an object-oriented, component-oriented programming language. C# provides language constructs to directly support these concepts, making C# a natural language in which to create and use software components. Since its origin, C# has added features to support new workloads and emerging software design practices. At its core, C# is an object-oriented language. You define types and their behavior.

Here's a brief description of C#

<https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>

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

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

Version Number	Release Date	Feature Count
1.0	1/2002	22
1.2	4/2003	2
2	11/2005	11
3	11/2007	10
4	4/2010	4
5	8/2012	3
6	6/2015	12
7.0	3/2017	12
7.1	8/2017	5
7.2	11/2017	6
7.3	5/2018	9
8.0	9/2019	14
9.0	11/2020	17
10.0	11/2021	16
11.0	11/2022	16

Here's a list of the features in each C# version.

<https://github.com/dotnet/csharp-lang/blob/main/Language-Version-History.md>

[https://en.wikipedia.org/wiki/C_Sharp_\(programming_language\)](https://en.wikipedia.org/wiki/C_Sharp_(programming_language))

Version Number	Release Date	Feature Count
1.0	1/2002	22
1.2	4/2003	2
2	11/2005	11
3	11/2007	10
4	4/2010	4
5	8/2012	3
6	6/2015	12
7.0	3/2017	12
7.1	8/2017	5
7.2	11/2017	6
7.3	5/2018	9
8.0	9/2019	14
9.0	11/2020	17
10.0	11/2021	16
11.0	11/2022	16

There's an interesting change in this history of language evolution – it happens between C# 5 and C# 6



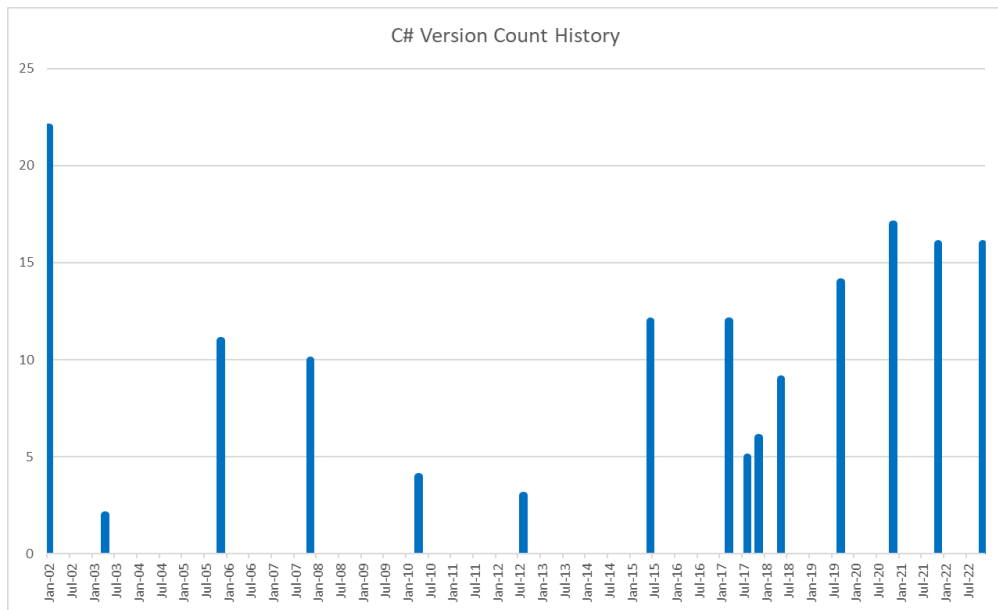
The .NET Compiler Platform

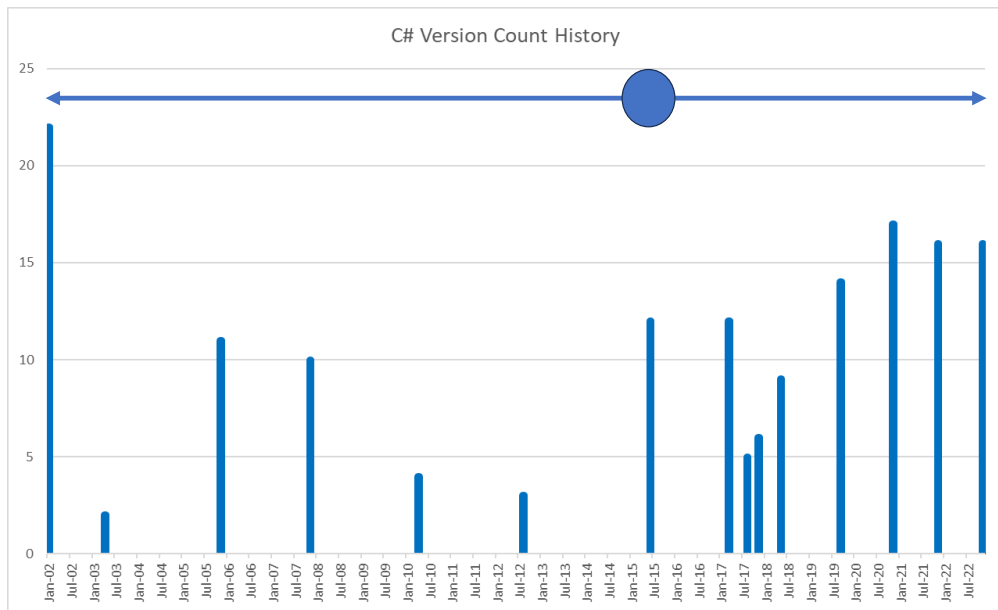
[gitter](#) [join chat](#) [12486 ONLINE](#)

Roslyn is the open-source implementation of both the C# and Visual Basic compilers with an API surface for building code analysis tools.

At that point, C# becomes an open-source, cross-platform language that compiles itself. The new compiler, code named Roslyn (officially “the Compiler API”), is written in C#, hosted in GitHub, and takes PRs from community members to improve the language across many different concerns (performance, reliability, etc.)

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





Before Roslyn
52 features in 10.5 years

After Roslyn
107 features in 7.5 years

C# has doubled the number of features in less time.

Before Roslyn
5 features a year

After Roslyn
14.25 features a year

It's almost tripled the amount of features released every year on average



It's interesting to note that before Roslyn, there were general concerns that C# (and .NET in general) was moving too slow and wasn't appealing to new developers - the pace was glacial.

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



Since Rolsyn, the consensus has flipped. Some developers feel like the pace is too quick and it's too hard to keep up.

<https://unsplash.com/photos/QUfxuCqdpH0>



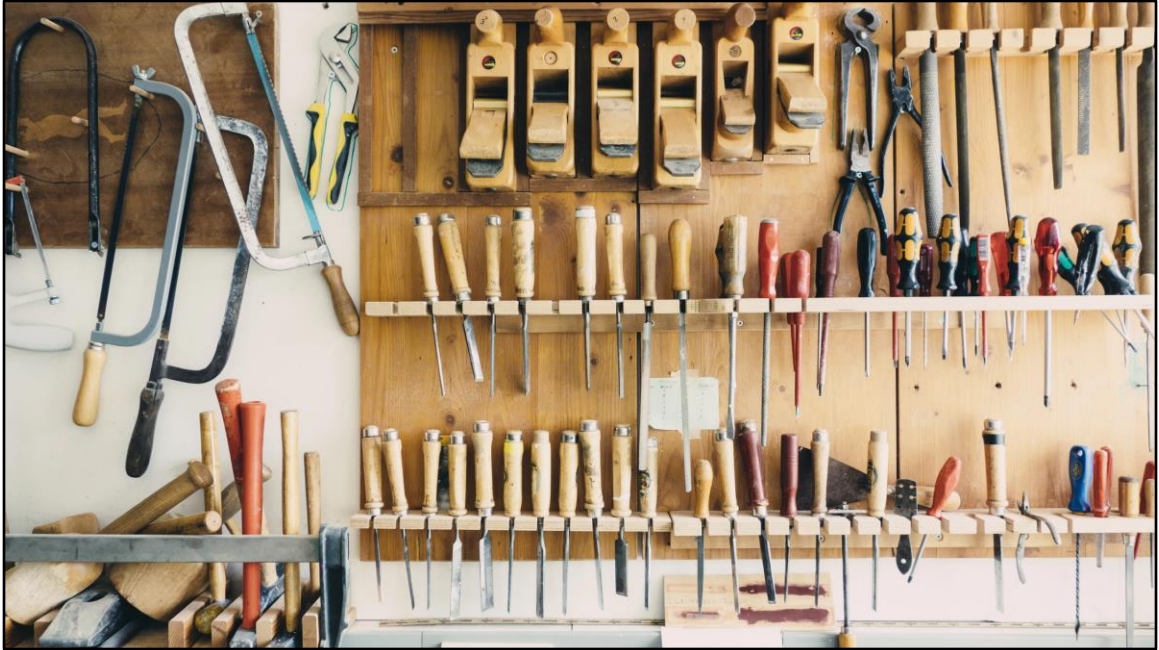
So, how do you keep up? Is it worth using new features? What strategies can you use?

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



Part of it is education. No matter what language you use, you have to spend time learning how the language works, not only at the current point in time, but also as the language evolves.

<https://www.pexels.com/photo/people-at-library-sitting-down-at-tables-757855/>



But you can also use a plethora of tools within IDEs to assist you in your journey. There are analyzers and refactoring that will light up and suggest changes you can make that use modern C# features.

<https://unsplash.com/photos/t5YUoHW6zRo>

Demo: Using Modern C# Features

Writing Modern C#

Let's go through C# that doesn't use modern features, and see how we can improve it

Features

`.editorconfig`

`Directory.Build.props`

`Directory.Packages.props`

`.csproj (SDK style)`

Features

Required
Properties

Tuples

Nullable
Reference
Types

Patterns

Raw String
Literals

Static Abstract
Members in
Interfaces

Target-Type
New

Object
Deconstruction

Records

Source
Generators

Call To Action

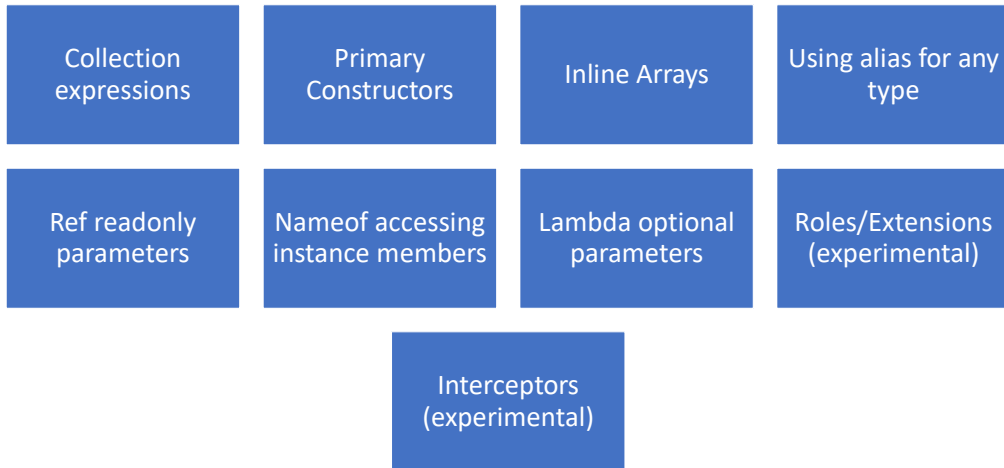
Writing Modern C#



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/IUY_3DvM__w

C# 12



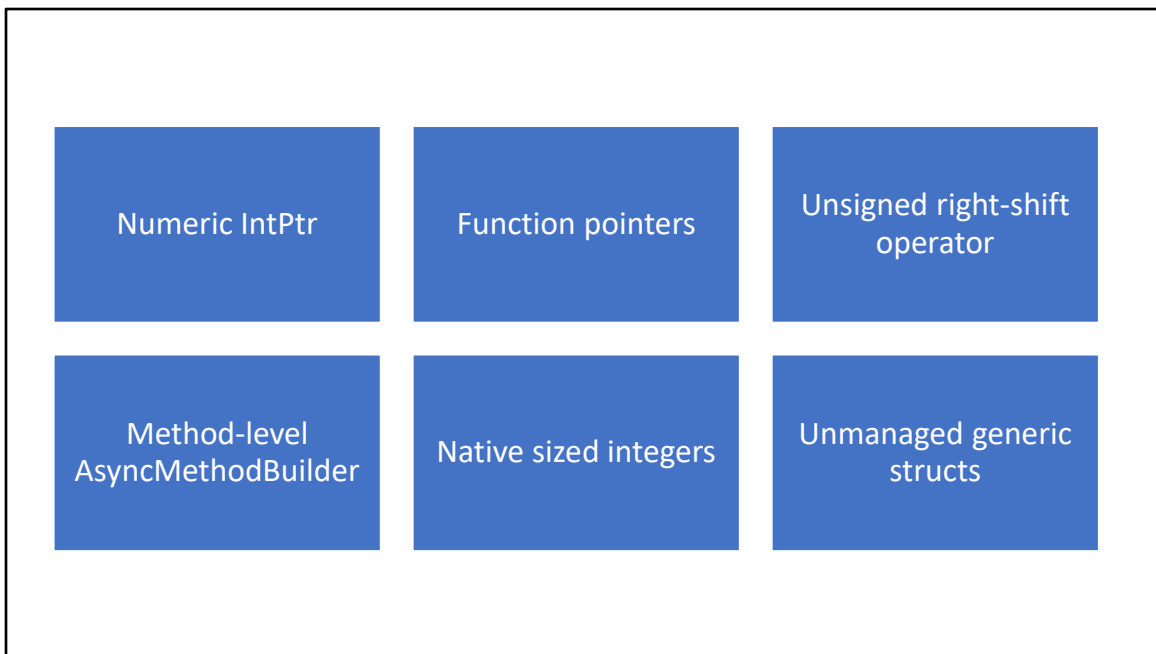
The feature trend isn't slowing down. C# 12, which will be released Nov. 2023, will have 7 new features, and 2 rather large experimental features to try.

<https://github.com/dotnet/csharp-lang/blob/main/Language-Version-History.md#c-120---net-8-and-visual-studio-2022-version-178>



Given that there doesn't seem to be any letup in this pace, the key is to keep studying.

https://unsplash.com/photos/QJDzYT_K8Xg



This is a sample of recent C# features that you may have never heard of. There are language features that are meant for infrequent usage or corner-case application performance scenarios.

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



Pick and choose which features are relevant to you. Not every feature is something that you'll use on a day-to-day basis.

<https://unsplash.com/photos/98MbUldcDJY>

Writing Modern C#

Jason Bock
Staff Software Engineer
Rocket Mortgage

Remember...

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

References

- * [C# Language standard](<https://github.com/dotnet/csharpstandard>)
- * [Features Added in C# Language Versions](<https://github.com/dotnet/csharplang/blob/main/Language-Version-History.md>)
- * [Language Feature Status](<https://github.com/dotnet/roslyn/blob/main/docs/Language%20Feature%20Status.md>)
- * [C# Guide](<https://docs.microsoft.com/en-us/dotnet/csharp/>)
- * [Language Version Planning](<https://github.com/dotnet/csharplang/projects/4>)
- * [Language Design Meetings](<https://github.com/dotnet/csharplang/tree/main/meetings>)