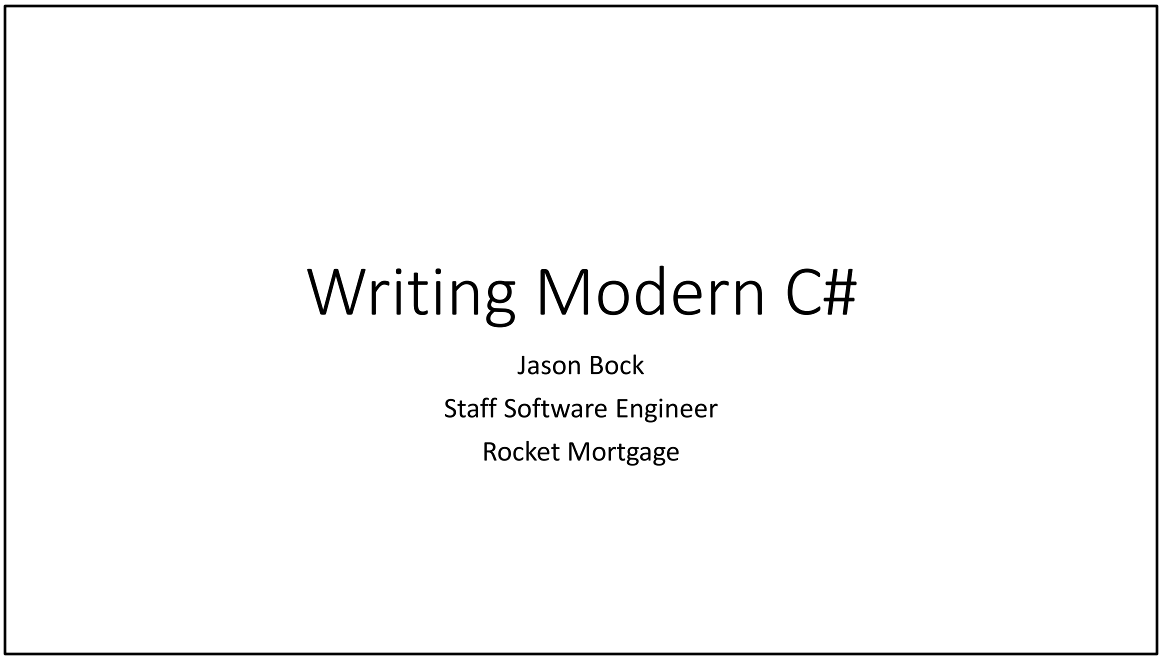
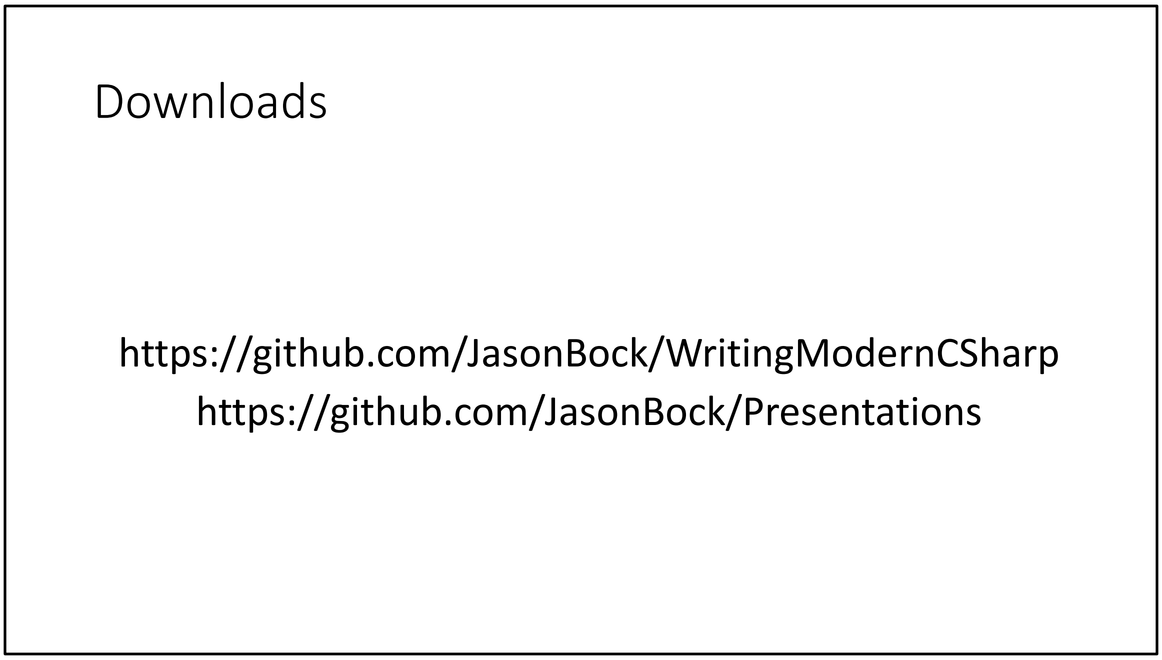


Writing Modern C#

Jason Bock
Staff Software Engineer
Rocket Mortgage

Personal Info

- https://jasonbock.net
- https://bsky.app/profile/jasonbock.net
- 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

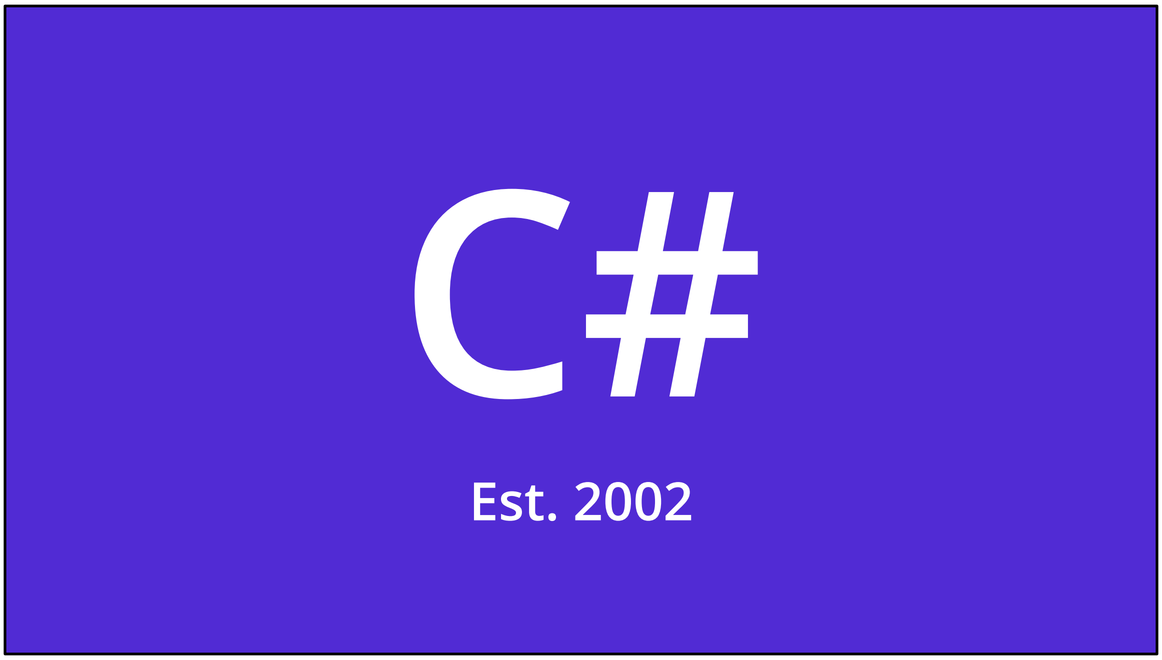
Language Evolution

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/



C#

Est. 2002

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

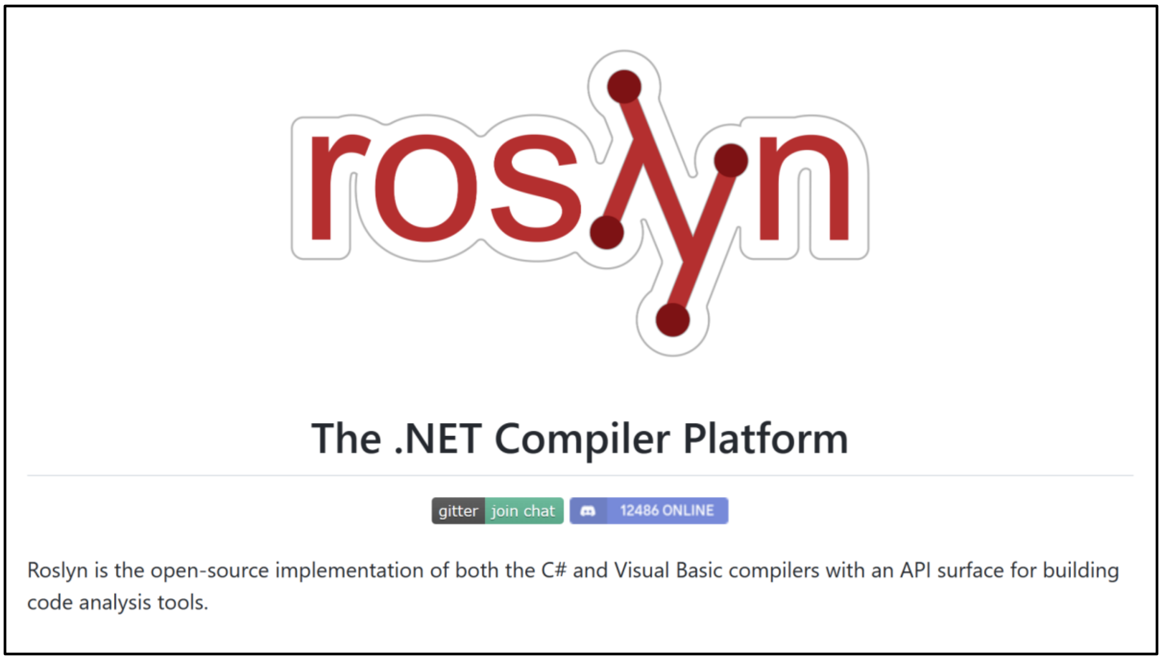
Version Number	Release Date	Feature Count
1.0	1/2002	22
1.2	4/2003	2
2	11/2005	11
3	11/2007	10
4	4/2010	4
5	8/2012	3
6	6/2015	12
7.0	3/2017	12
7.1	8/2017	5
7.2	11/2017	6
7.3	5/2018	9
8.0	9/2019	14
9.0	11/2020	17
10.0	11/2021	16
11.0	11/2022	16
12.0	11/2023	7
13.0	11/2024	10

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

https://github.com/dotnet/csharplang/blob/main/Language-Version-History.md
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
12.0	11/2023	7
13.0	11/2024	10

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

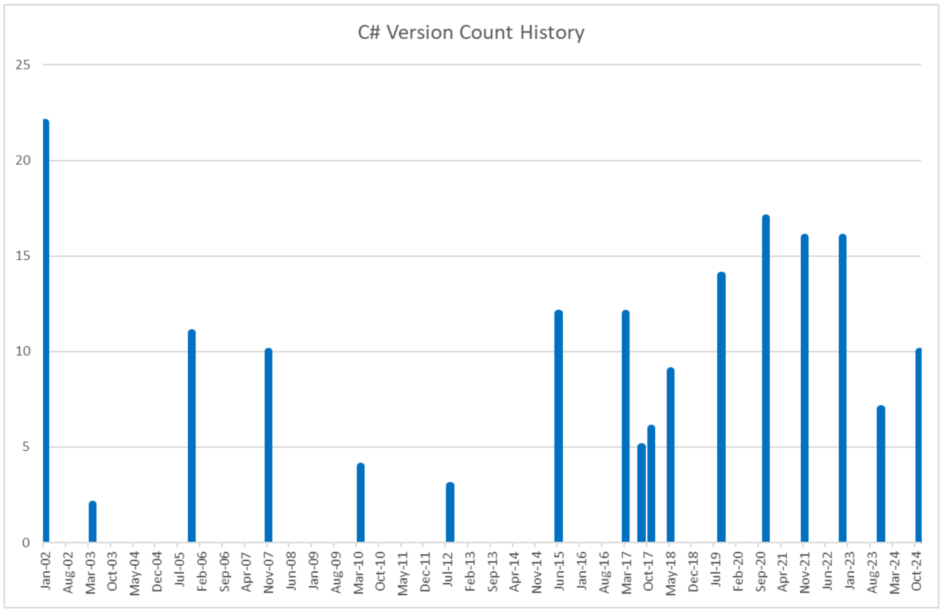


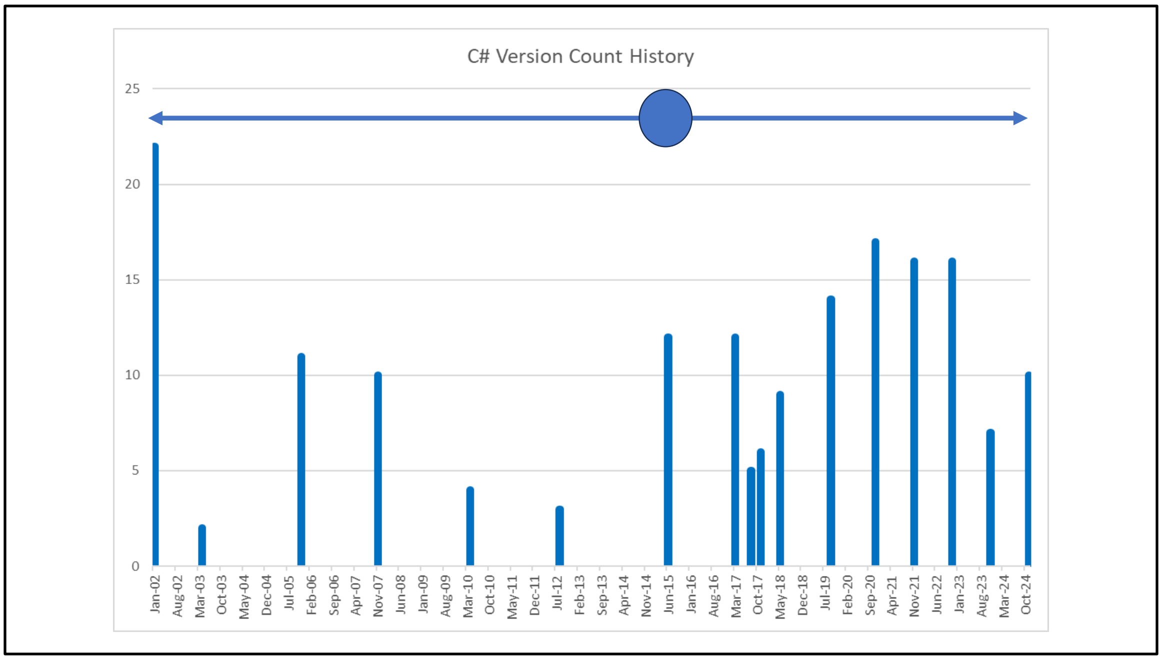
The .NET Compiler Platform

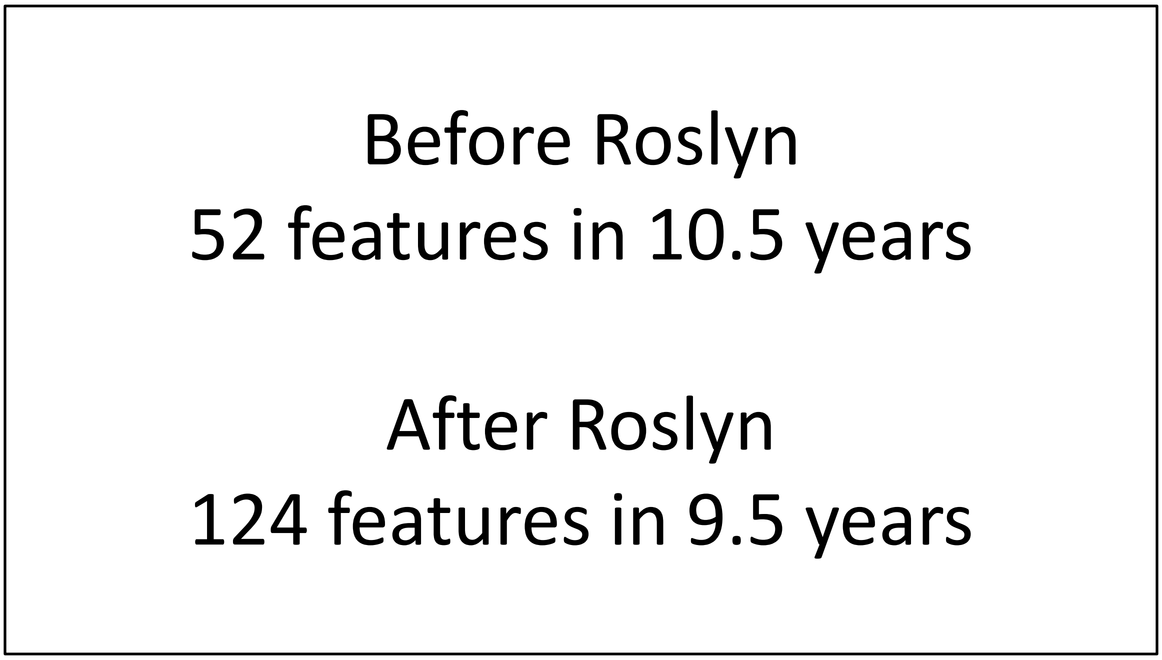
Roslyn is the open-source implementation of both the C# and Visual Basic compilers with an API surface for building code analysis tools.

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


C# Version Count History
25
20
15
10
5
0
Jan-02
Aug-02
Mar-03
Oct-03
May-04
Dec-04
Jul-05
Feb-06
Sep-06
Apr-07
Nov-07
Jun-08
Jan-09
Aug-09
Mar-10
Oct-10
May-11
Dec-11
Jul-12
Feb-13
Sep-13
Apr-14
Nov-14
Jun-15
Jan-16
Aug-16
Mar-17
Oct-17
May-18
Dec-18
Jul-19
Feb-20
Sep-20
Apr-21
Nov-21
Jun-22
Jan-23
Aug-23
Mar-24
Oct-24


C# Version Count History
25
20
15
10
5
0
Jan-02
Aug-02
Mar-03
Oct-03
May-04
Dec-04
Jul-05
Feb-06
Sep-06
Apr-07
Nov-07
Jun-08
Jan-09
Aug-09
Mar-10
Oct-10
May-11
Dec-11
Jul-12
Feb-13
Sep-13
Apr-14
Nov-14
Jun-15
Jan-16
Aug-16
Mar-17
Oct-17
May-18
Dec-18
Jul-19
Feb-20
Sep-20
Apr-21
Nov-21
Jun-22
Jan-23
Aug-23
Mar-24
Oct-24



Before Roslyn
52 features in 10.5 years

After Roslyn
124 features in 9.5 years

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

Before Roslyn
5 features a year

After Roslyn
13 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.

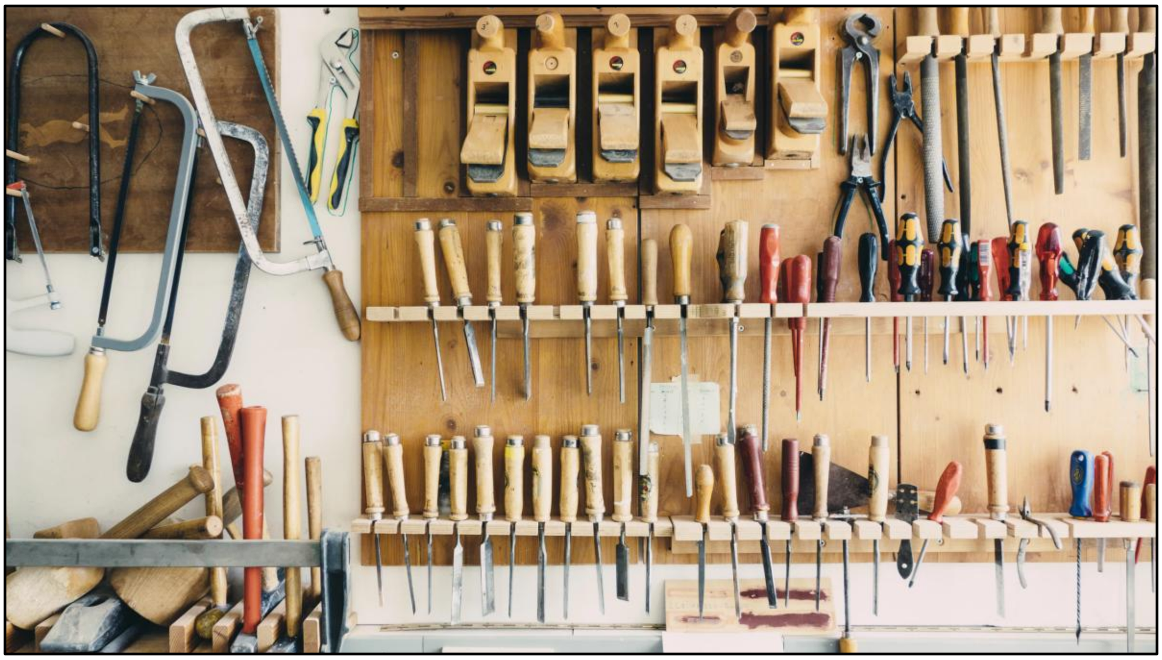
https://unsplash.com/photos/QUfxuCqdpH0

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

https://unsplash.com/photos/ufgOEVZuHgM

Part of it is education. No matter what language you use, you must 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/



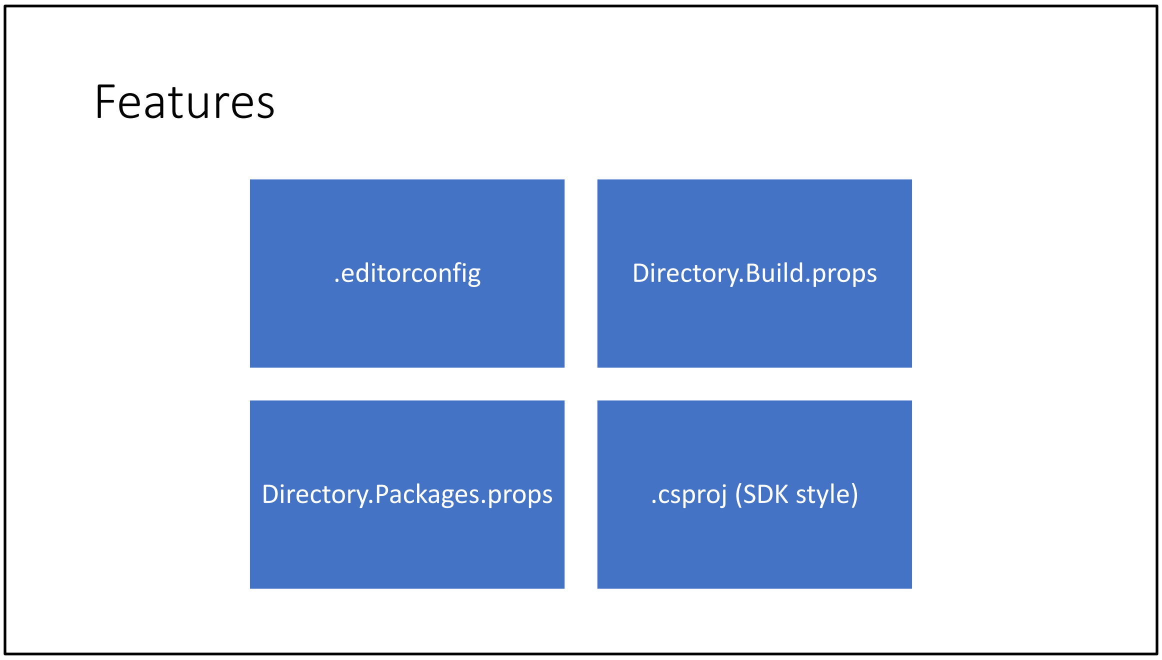
But you can also use a plethora of tools within IDEs to assist you in your journey. There are analyzers and refactoring that will light up and suggest changes you can make that use modern C# features.

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

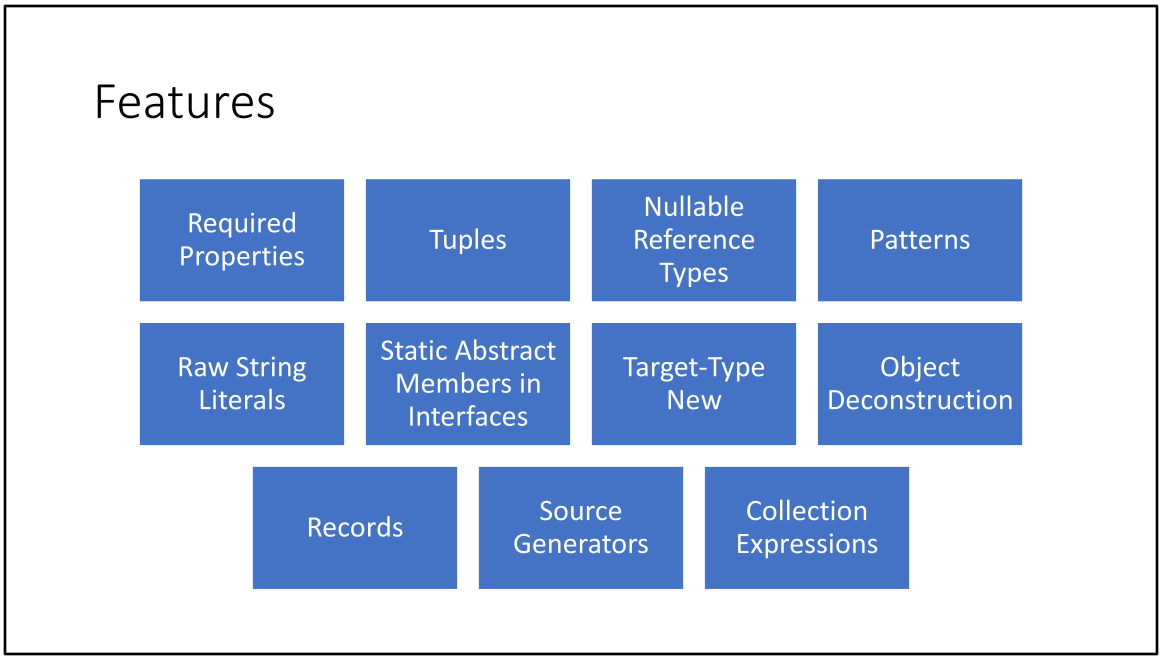
.editorconfig

Directory.Build.props

Directory.Packages.props

.csproj (SDK style)

Here are some things to make it easier to stay on top of new features as well as easily change when new features come out.

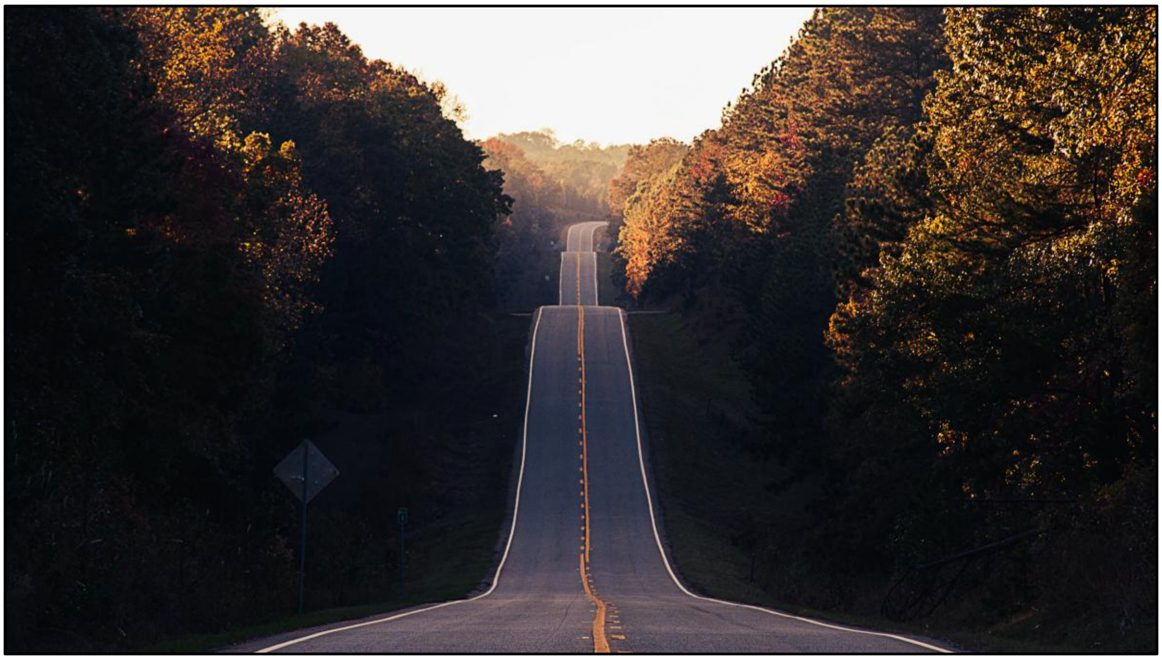


Features

- Required Properties
- Tuples
- Nullable Reference Types
- Patterns
- Raw String Literals
- Static Abstract Members in Interfaces
- Target-Type New
- Object Deconstruction
- Records
- Source Generators
- Collection Expressions

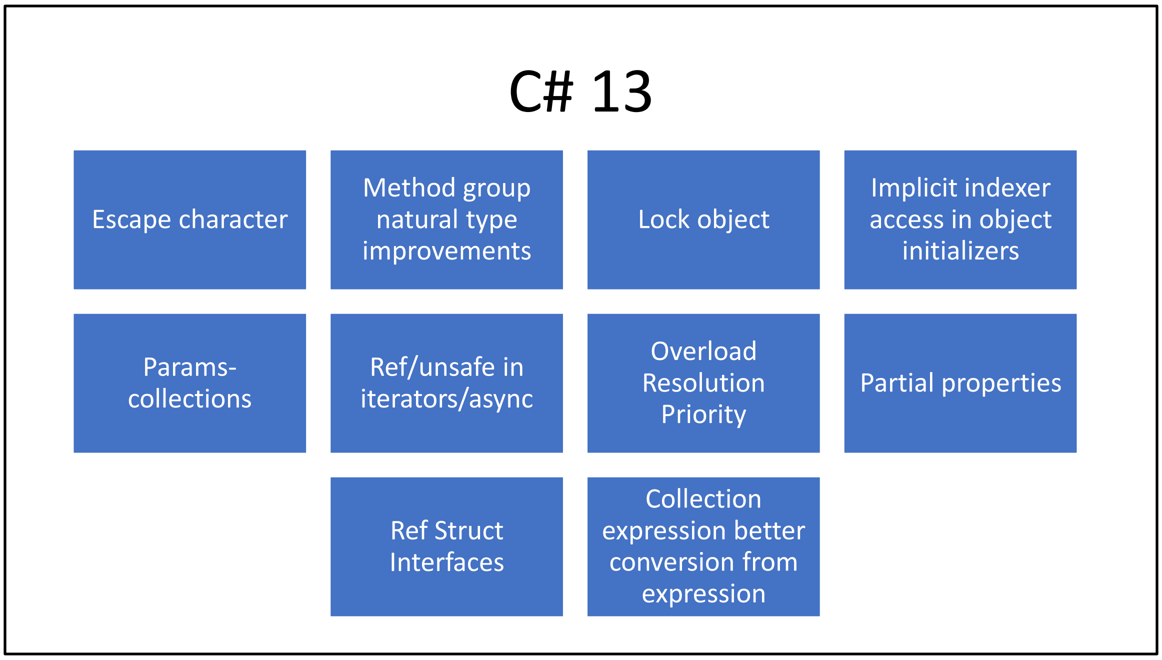
Call To Action

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

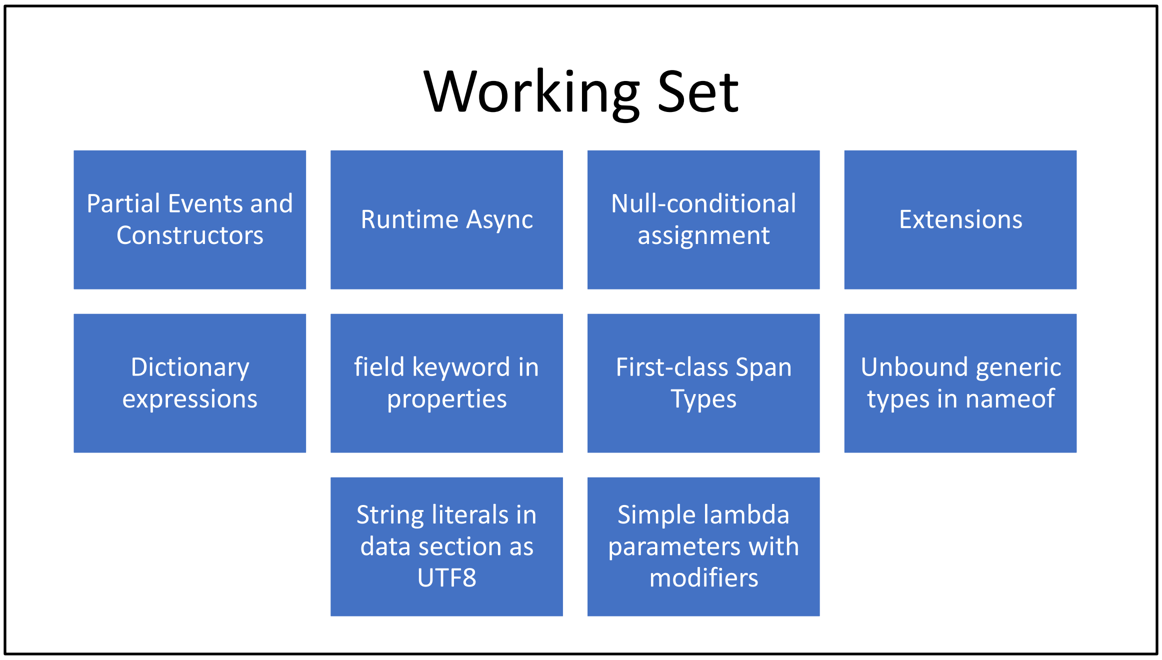


C# 13

- Escape character
- Method group natural type improvements
- Lock object
- Implicit indexer access in object initializers
- Params-collections
- Ref/unsafe in iterators/async
- Overload Resolution Priority
- Partial properties
- Ref Struct Interfaces
- Collection expression better conversion from expression

The feature trend isn’t slowing down. C# 13, which was released Nov. 2024, has 10 new features.

https://github.com/dotnet/roslyn/blob/main/docs/Language%20Feature%20Status.md#working-set

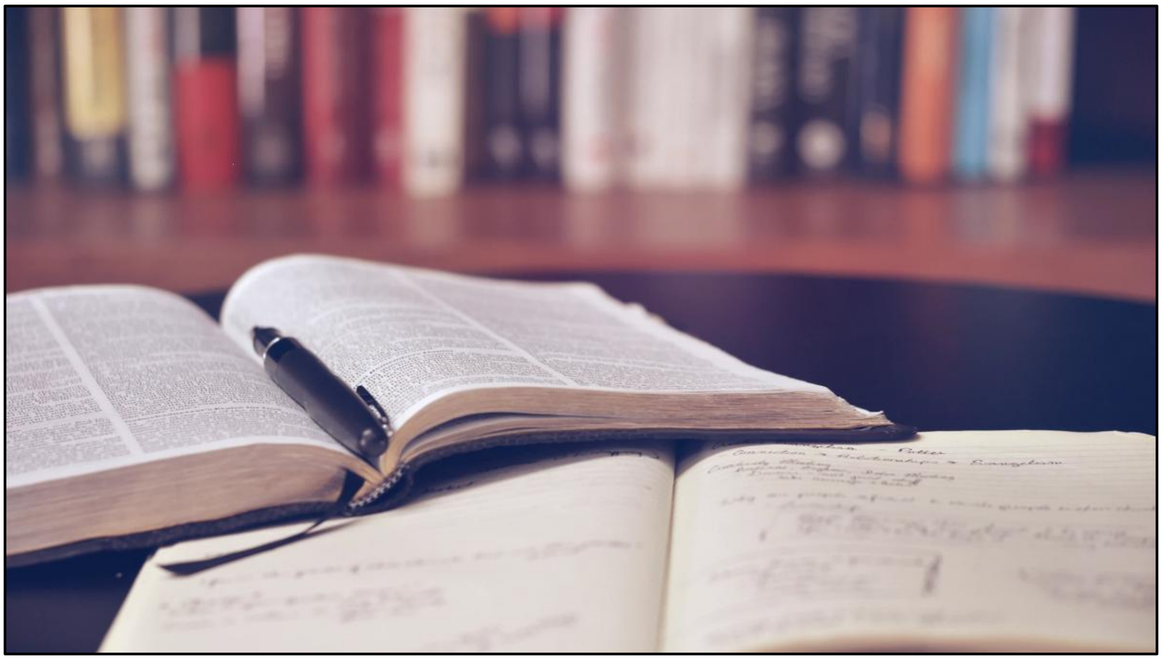


Working Set

- Partial Events and Constructors
- Runtime Async
- Null-conditional assignment
- Extensions
- Dictionary expressions
- field keyword in properties
- First-class Span Types
- Unbound generic types in nameof
- String literals in data section as UTF8
- Simple lambda parameters with modifiers

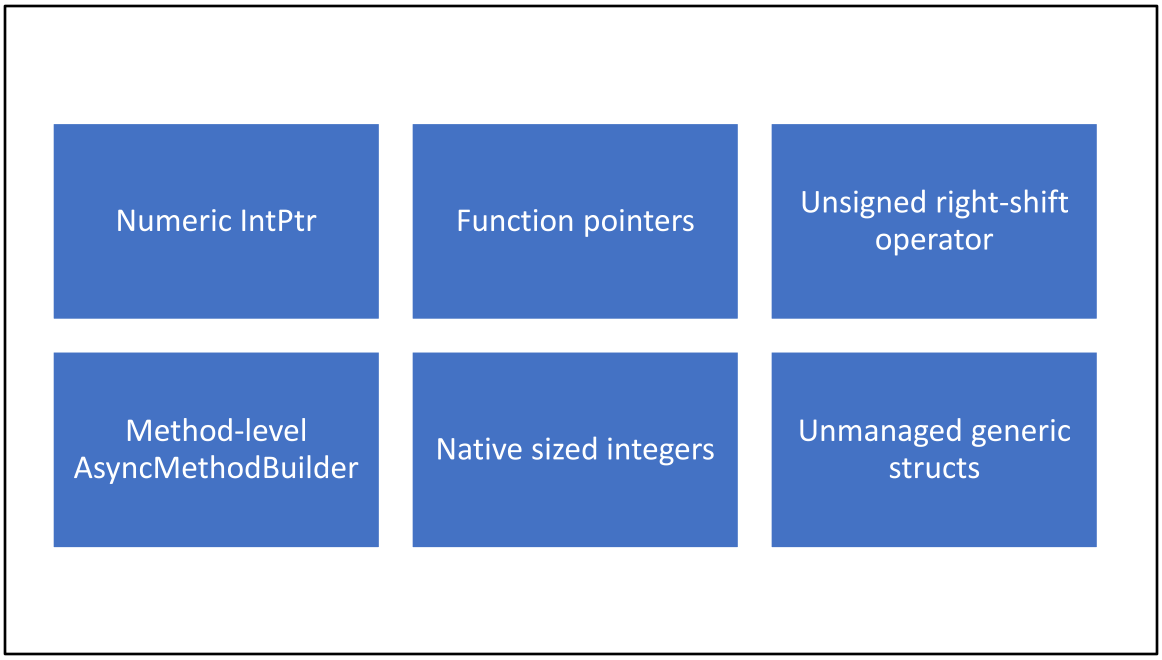
There's also a working set of features that may happen at some point in the future.

https://github.com/dotnet/roslyn/blob/main/docs/Language%20Feature%20Status.md#working-set



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



- Numeric IntPtr
- Function pointers
- Unsigned right-shift operator
- Method-level AsyncMethodBuilder
- Native sized integers
- Unmanaged generic structs

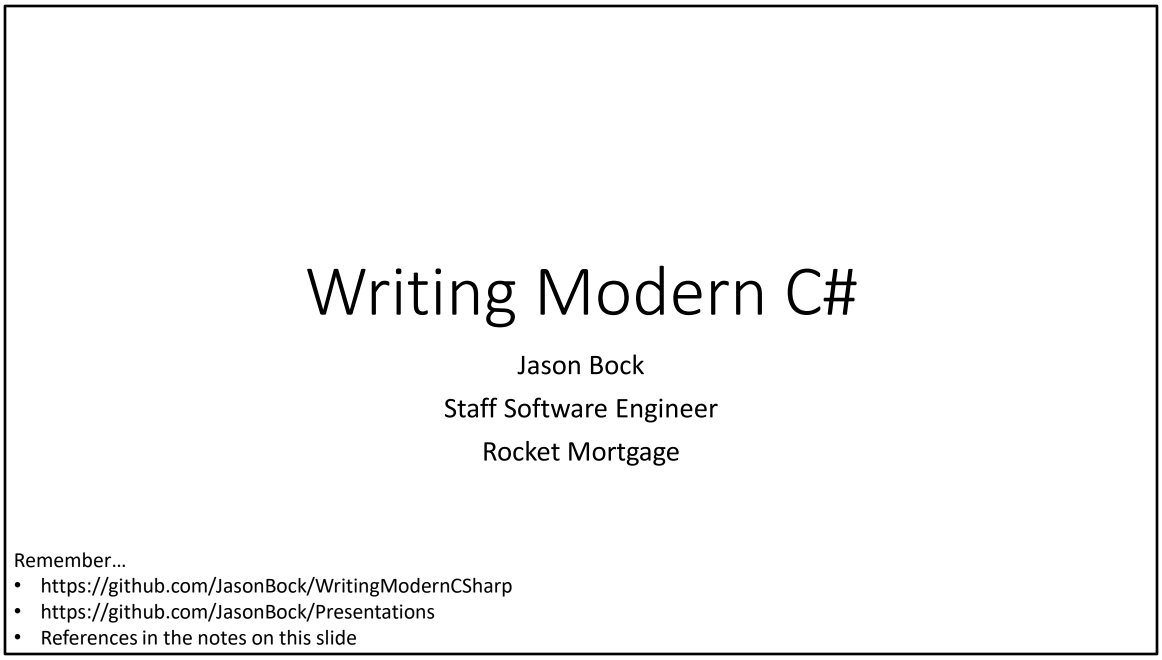
This is a sample of recent C# features that you may have never heard of. There are language features that are meant for infrequent usage or corner-case application performance scenarios.

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

GRAMMES
PORTEE MINIMUM
PORTEE MAXIMUM 2 KILOG
AVERY
Societe Belge des Balances & Bascules
RUE DE L'INTENDANT, 43 - BRUXELLES

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)
* C# 12
    * [Dark side of the primary constructors in C# 12](https://mareks-082.medium.com/dark-side-of-the-primary-constructors-in-c-12-b9d75855d4df)
* C# 11
    * [Welcome to C# 11](https://devblogs.microsoft.com/dotnet/welcome-to-csharp-11/)
    * [Regular Expression Improvements in .NET 7](https://devblogs.microsoft.com/dotnet/regular-expression-improvements-in-dotnet-7/)
    * [What's new in C# 11](https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-11)
    * [.NET 7 Preview 5 - Generic Math](https://devblogs.microsoft.com/dotnet/dotnet-7-generic-math/)
    * [Twelve C# 11 Features](https://blog.okyrylchuk.dev/twelve-csharp-11-features)
* C# 10
    * [What's New in C# 10: Take Control of Interpolated String Handling](http://dontcodetired.com/blog/post/Whats-New-in-C-10-Take-Control-of-Interpolated-String-Handling)
    * [Dissecting Interpolated Strings Improvements in C# 10](https://sergeyteplyakov.github.io/Blog/c%2310/2021/11/08/Dissecing-Interpolated-Strings-Improvements-In-CSharp-10.html)
    * [C# 10 - `record struct` Deep Dive & Performance Implications](https://nietras.com/2021/06/14/csharp-10-record-struct/)
    * [String Interpolation in C# 10 and .NET 6](https://devblogs.microsoft.com/dotnet/string-interpolation-in-c-10-and-net-6/)
* 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-corert-pure-magic-in-my.html)
    * [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/)
        * [Incremental Generators
Cookbook](https://github.com/dotnet/roslyn/blob/main/docs/featu
res/incremental-generators.cookbook.md)
* C# 8
    * [Embracing nullable reference
types](https://devblogs.microsoft.com/dotnet/embracing-
nullable-reference-types/)
    * [C# 8 default interface
methods](https://developers.redhat.com/blog/2020/03/03/c-8-
default-interface-methods/)
    * 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)
    * [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/)
    * [C# 8 Pattern Matching
Enhancements](https://www.infoq.com/news/2019/06/CSharp-8-
Pattern-Matching/)
    * [Async Streams with
C#8](https://csharp.christiannagel.com/2019/03/20/asyncstreams/
)
    * [Do more with patterns in C#
8.0](https://devblogs.microsoft.com/dotnet/do-more-with-
patterns-in-c-8-0/)
* C# 7
    * [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/)
    * [Performance traps of ref locals and ref returns in C#](https://devblogs.microsoft.com/premier-developer/performance-traps-of-ref-locals-and-ref-returns-in-c/)
    * [First steps with nullable reference types](https://codeblog.jonskeet.uk/2018/04/21/first-steps-with-nullable-reference-types/)
    * [Using In Parameter Modifier - C# 7.2](https://codewala.net/2018/03/27/using-in-parameter-modifier-c-7-2/)
    * [C# 7 Series](https://learn.microsoft.com/en-us/archive/blogs/mazhou/c-7-series-part-10-spant-and-universal-memory-management)
    * [Herding Nulls and Other C# Stories From the Future](https://www.infoq.com/presentations/c-sharp-future)
    * [Dissecting the local functions in C# 7](https://devblogs.microsoft.com/premier-developer/dissecting-the-local-functions-in-c-7/)
    * [Removing Switch-Case Statement and using Pattern Matching in C#](https://rubikscode.net/2022/07/18/abolishing-switch-case-and-pattern-matching-in-c/)
* C# 6
    * [C# : How C# 6.0 Simplifies, Clarifies and Condenses Your Code](https://learn.microsoft.com/en-us/archive/msdn-magazine/2014/special-issue/csharp-how-csharp-6-0-simplifies-clarifies-and-condenses-your-code)
    * C# 6.0: Language features and its internals
        * [Part 1](http://www.abhisheksur.com/2015/03/c-60-language-features-and-its.html)
        * [Part 2](http://www.abhisheksur.com/2015/03/c-60-language-features-and-its_11.html)
    * [Customizing string interpolation in C#6](http://www.thomaslevesque.com/2015/02/24/customizing-string-interpolation-in-c-6/)
    * [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)
```