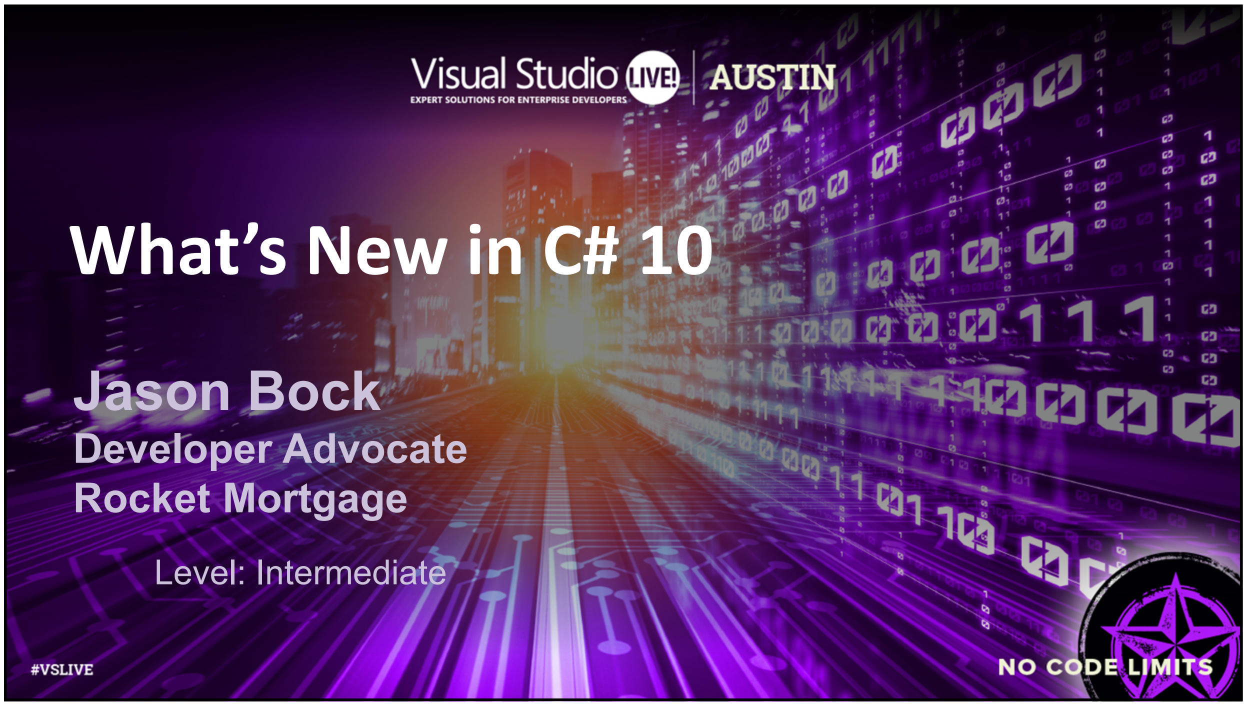
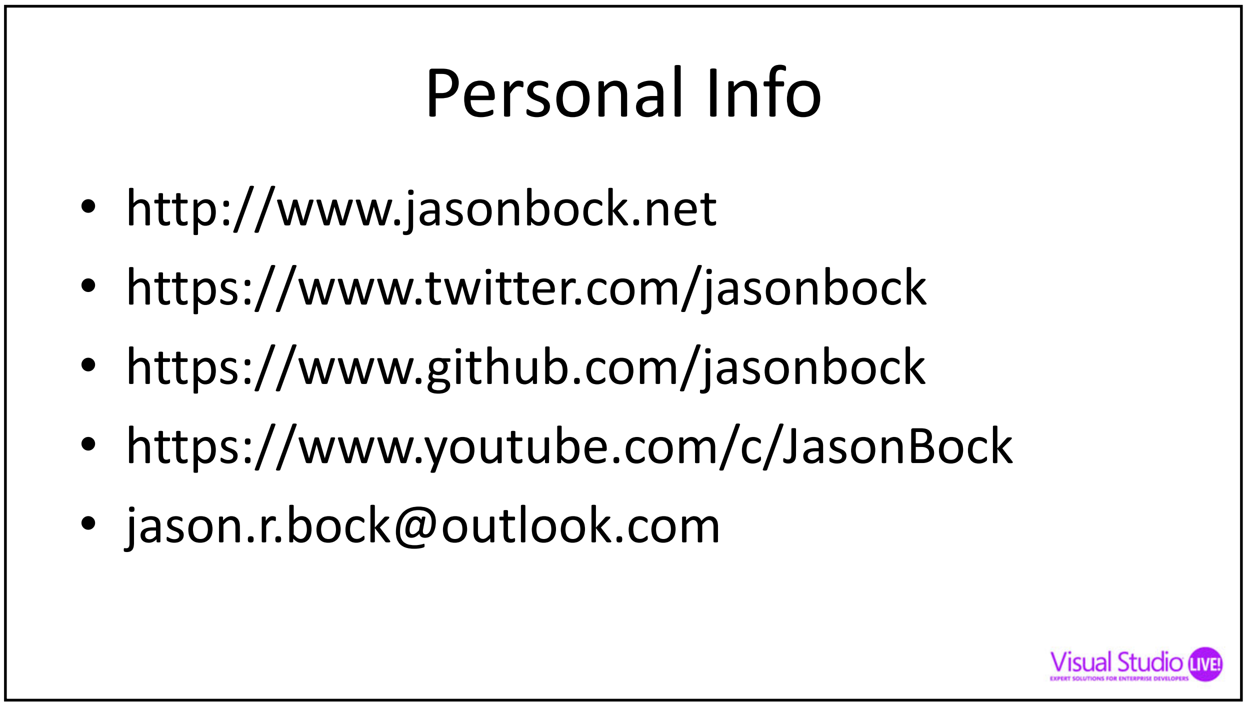


What's New in C# 10

Jason Bock
Developer Advocate
Rocket Mortgage

Level: Intermediate

Visual Studio LIVE! | AUSTIN
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
#VSLIVE
NO CODE LIMITS



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

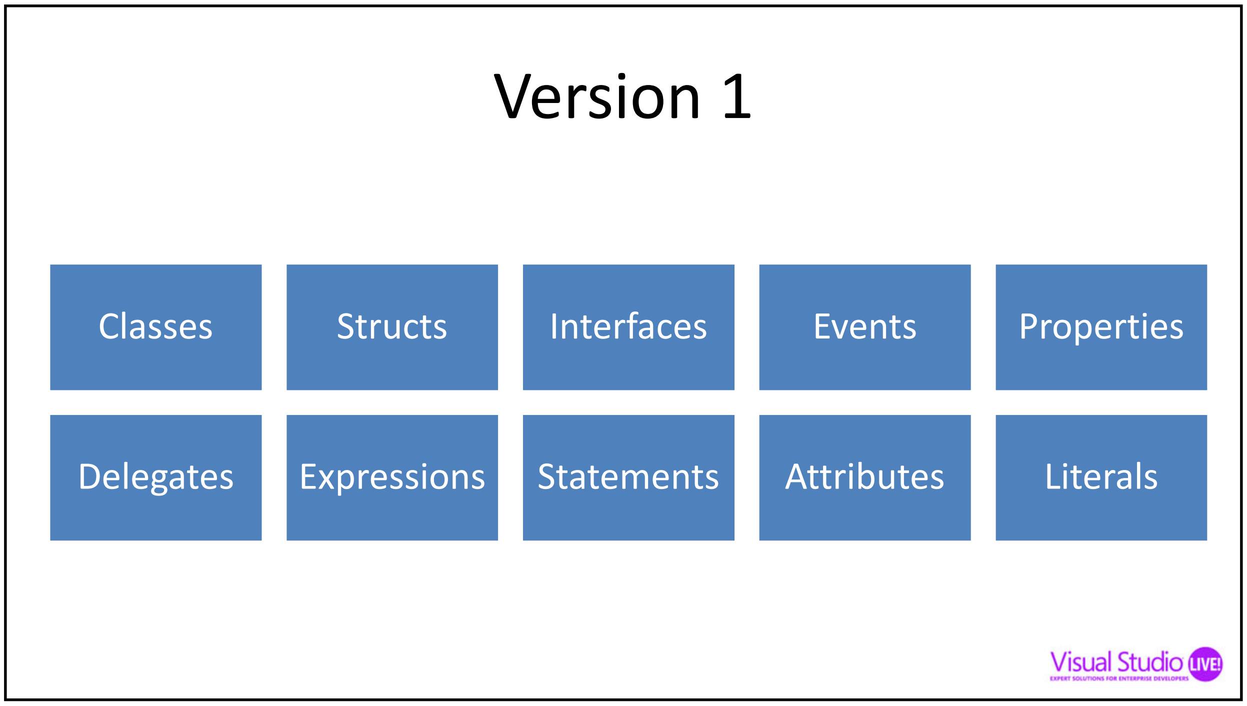
Remember…
https://github.com/JasonBock/WhatsNewInCSharp10
https://github.com/JasonBock/Presentations

What's New in C# 10

LANGUAGE EVOLUTION

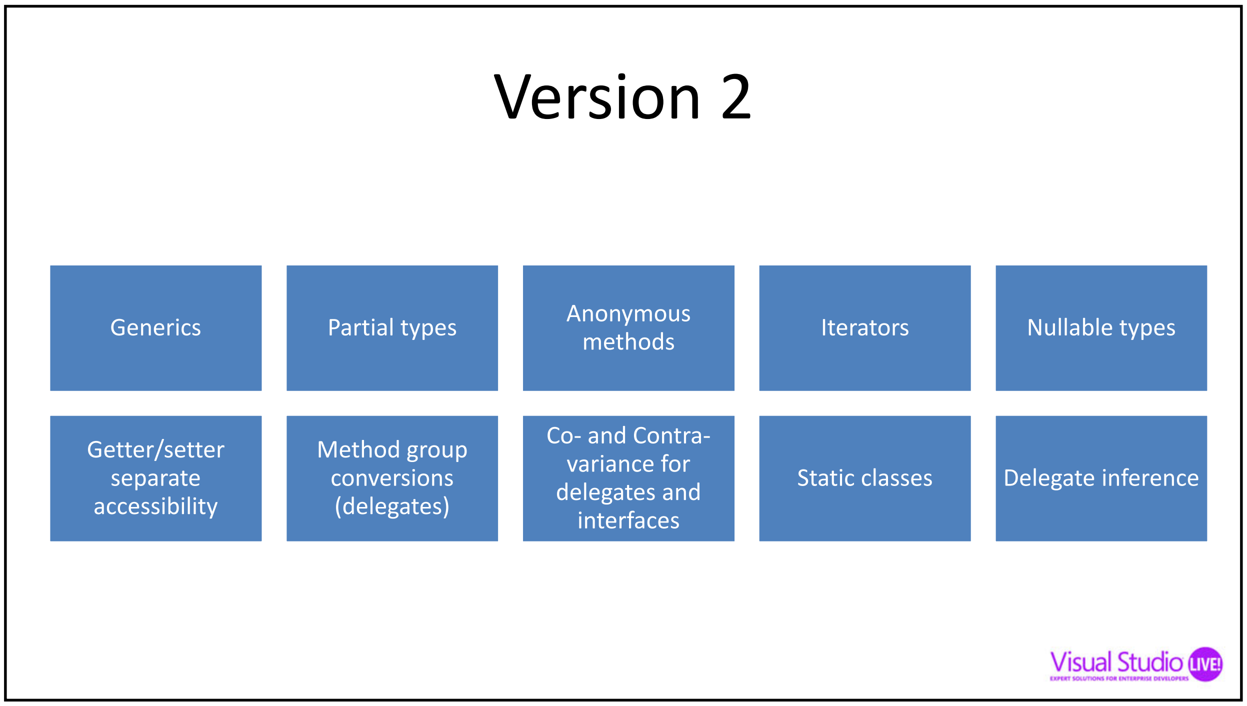
C#

Est. 2002



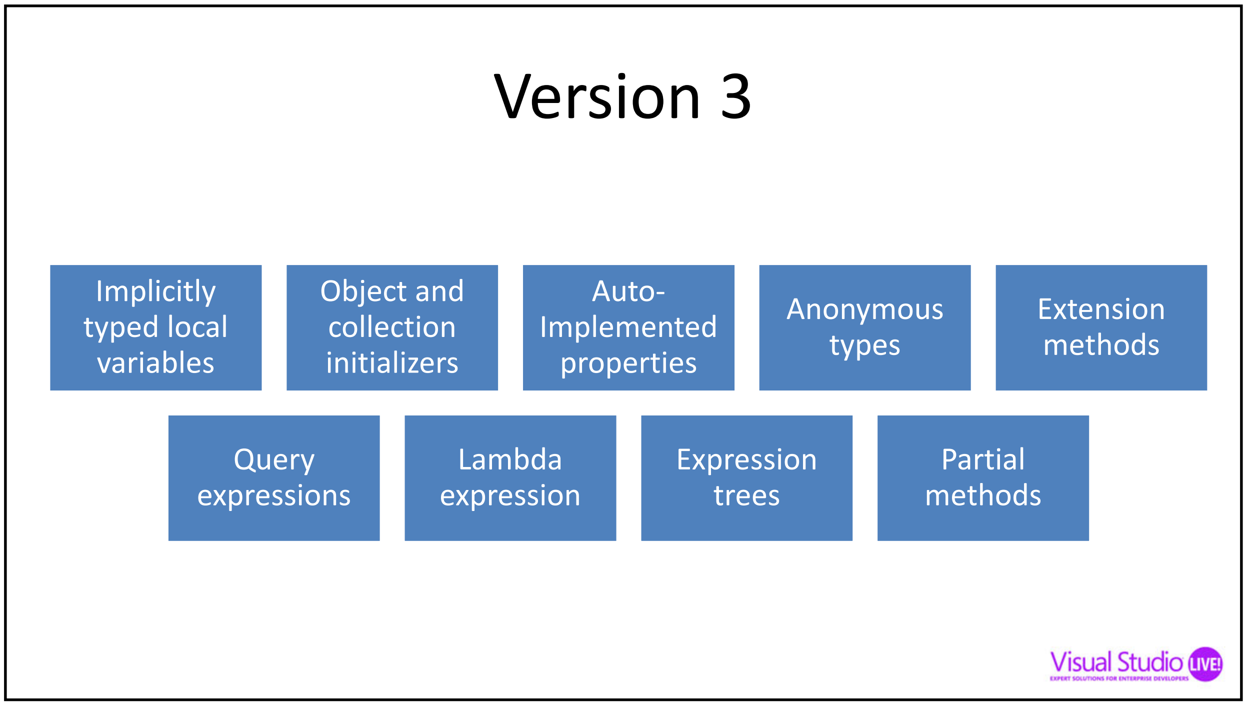
Version 1

- Classes
- Structs
- Interfaces
- Events
- Properties
- Delegates
- Expressions
- Statements
- Attributes
- Literals



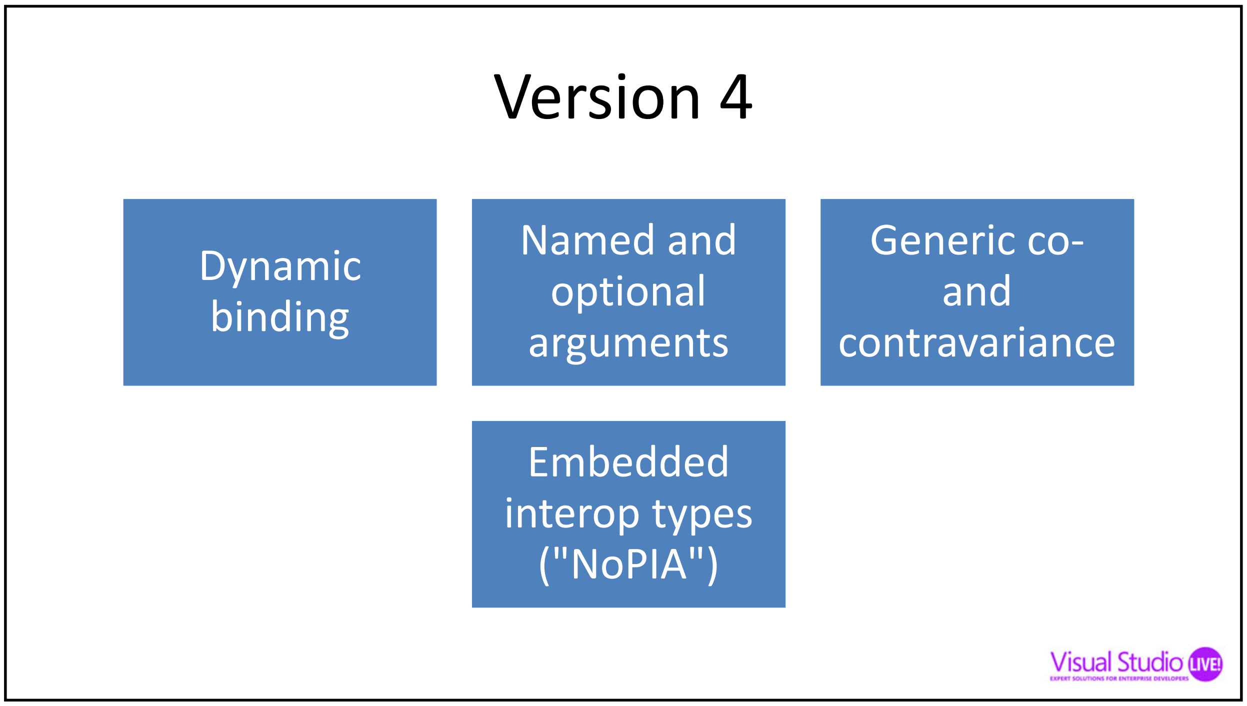
Version 2

- 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 3

- Implicitly typed local variables
- Object and collection initializers
- Auto-Implemented properties
- Anonymous types
- Extension methods
- Query expressions
- Lambda expression
- Expression trees
- Partial methods



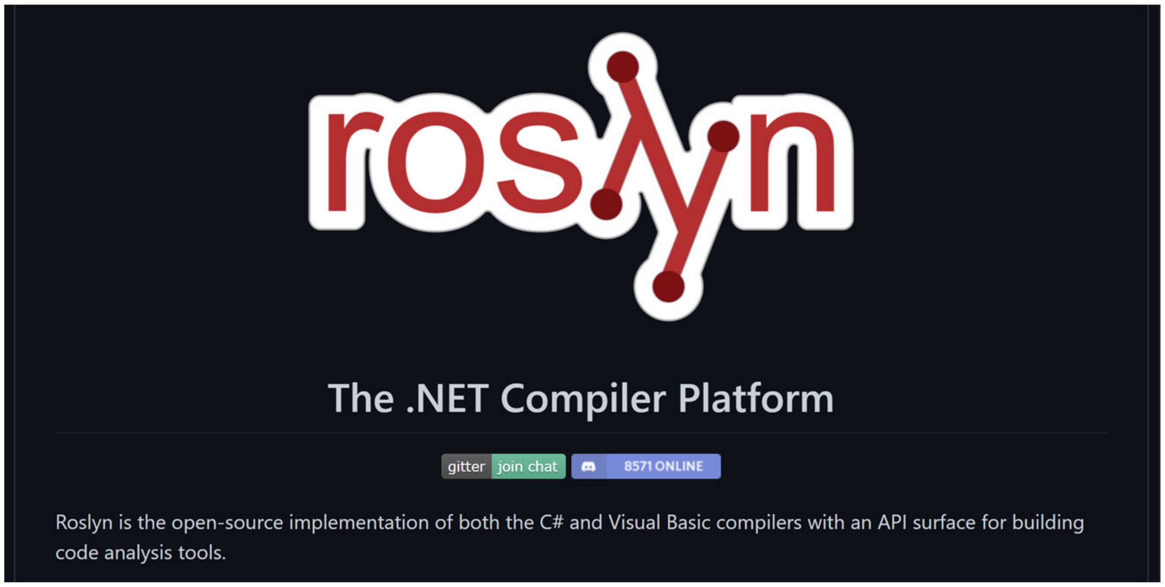
Version 4

- Dynamic binding
- Named and optional arguments
- Generic co- and contravariance
- Embedded interop types ("NoPIA")

Version 5

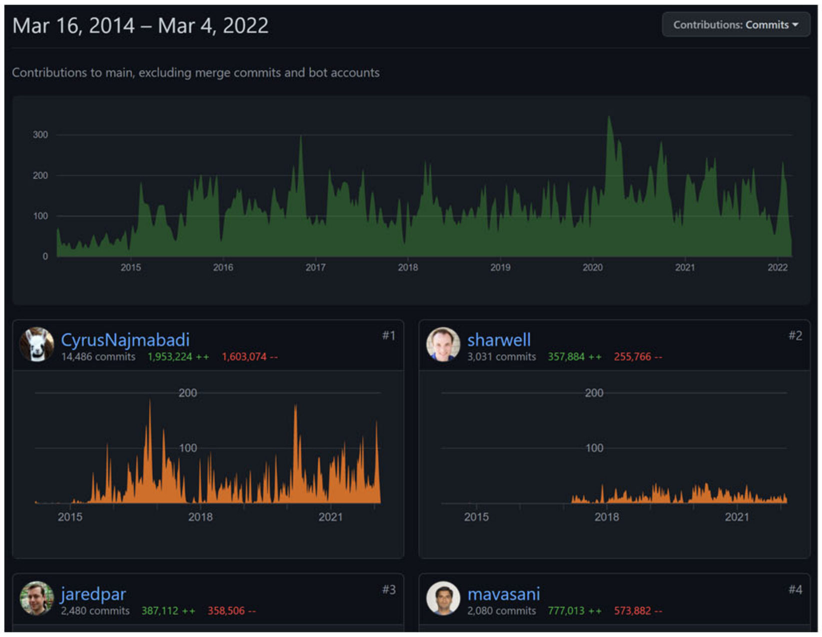
Asynchronous methods

Caller info attributes



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.


Mar 16, 2014 – Mar 4, 2022

Contributions: Commits

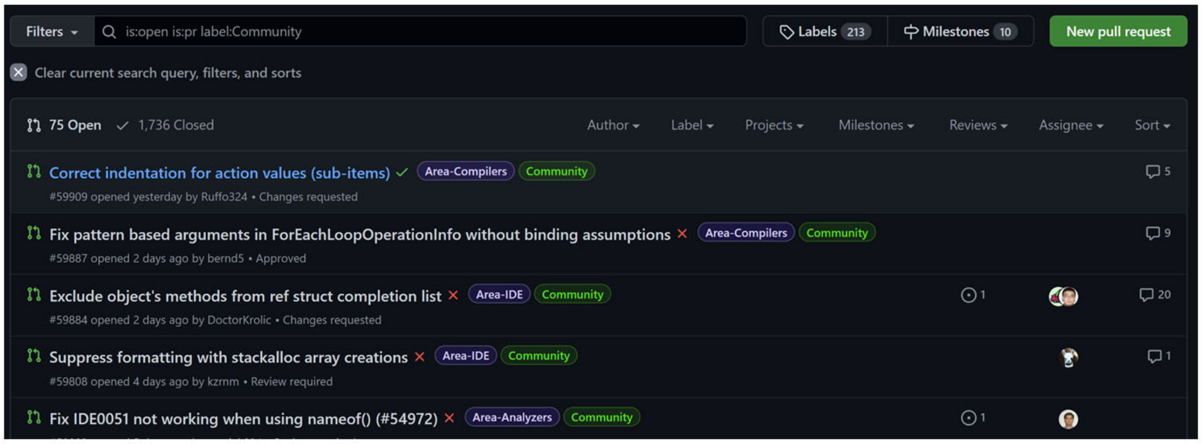
Contributions to main, excluding merge commits and bot accounts

CyrusNajmabadi #1
14,486 commits 1,953,224 ++ 1,603,074 --

sharwell #2
3,031 commits 357,884 ++ 255,766 --

jaredpar #3
2,480 commits 387,112 ++ 358,506 --

mavasani #4
2,080 commits 777,013 ++ 573,882 --


Filters is:open is:pr label:Community Labels 213 Milestones 10 New pull request

Clear current search query, filters, and sorts

75 Open ✓ 1,736 Closed Author Label Projects Milestones Reviews Assignee Sort

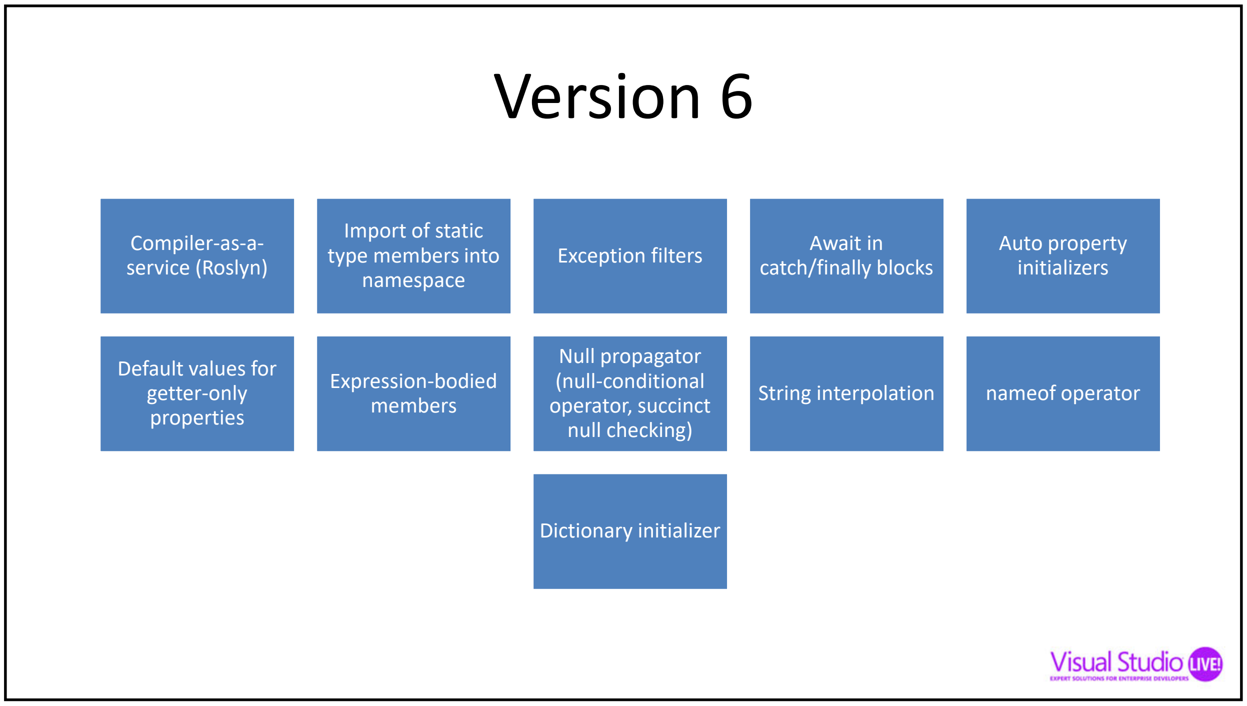
Correct indentation for action values (sub-items) ✓ Area-Compilers Community 5
#59909 opened yesterday by Ruffo324 • Changes requested

Fix pattern based arguments in ForEachLoopOperationInfo without binding assumptions ✗ Area-Compilers Community 9
#59887 opened 2 days ago by bernd5 • Approved

Exclude object's methods from ref struct completion list ✗ Area-IDE Community 1 20
#59884 opened 2 days ago by DoctorKrolic • Changes requested

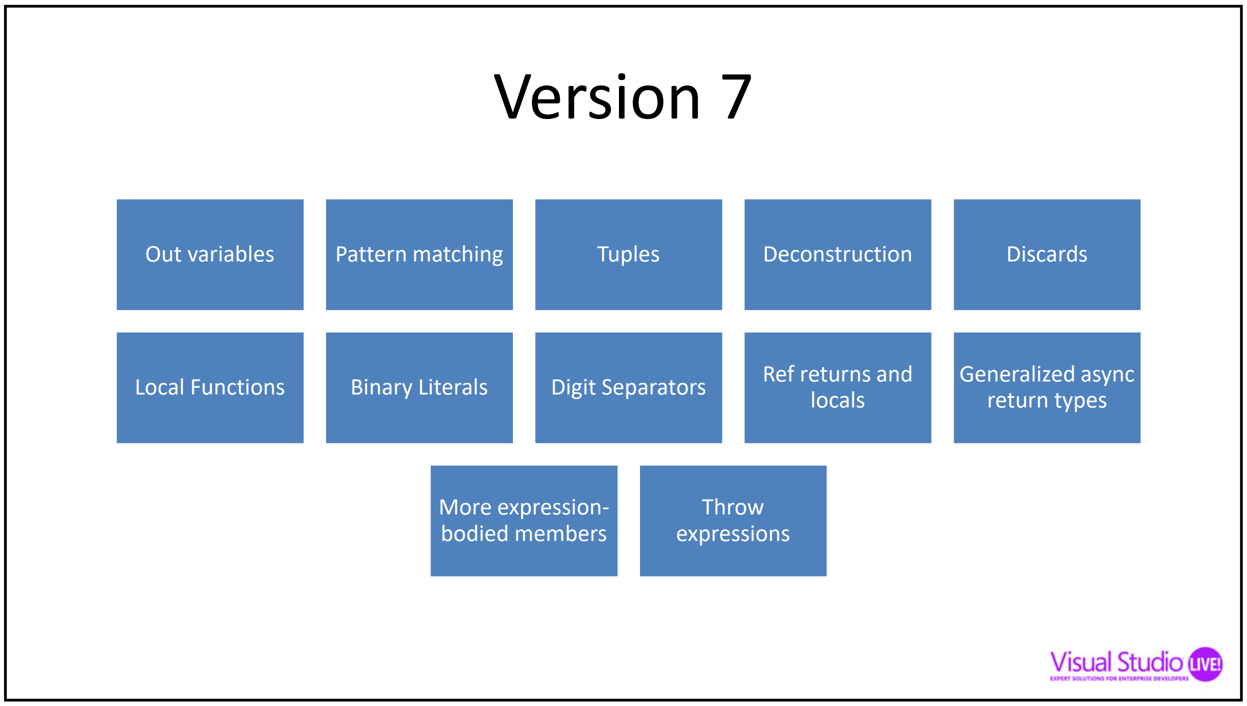
Suppress formatting with stackalloc array creations ✗ Area-IDE Community 1
#59808 opened 4 days ago by kzrnm • Review required

Fix IDE0051 not working when using nameof() (#54972) ✗ Area-Analyzers Community 1



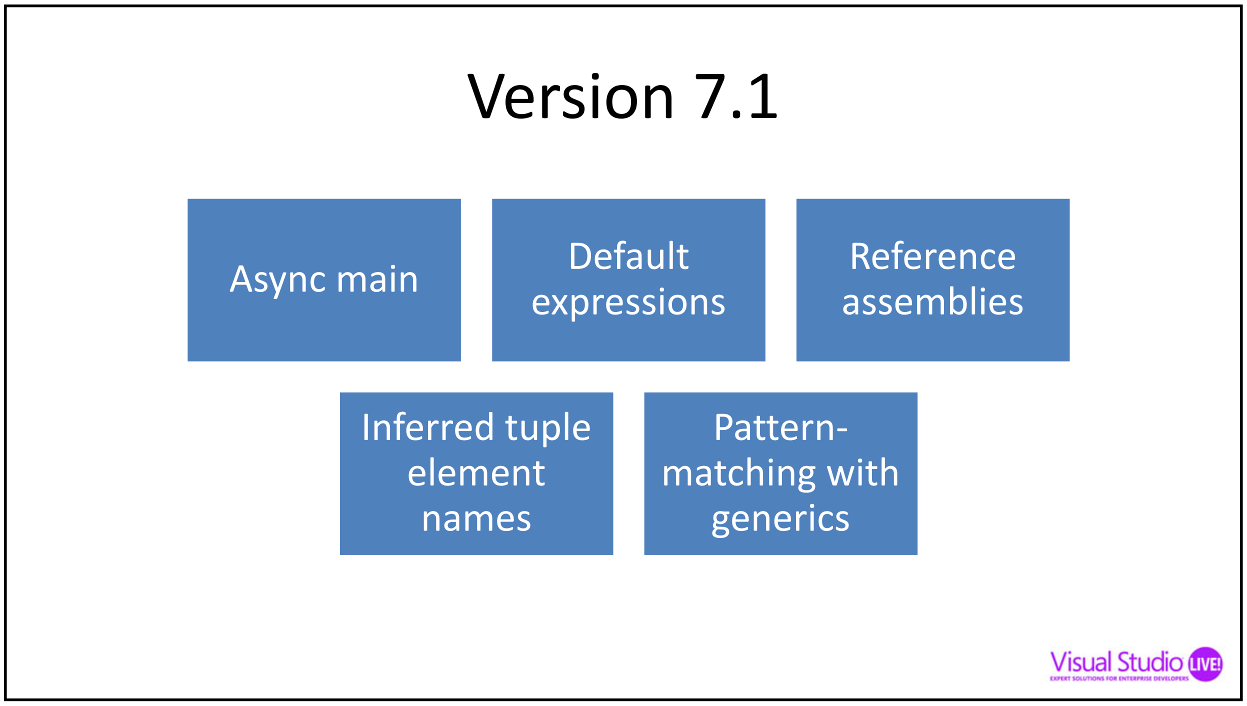
Version 6

- Compiler-as-a-service (Roslyn)
- Import of static type members into namespace
- Exception filters
- Await in catch/finally blocks
- Auto property initializers
- Default values for getter-only properties
- Expression-bodied members
- Null propagator (null-conditional operator, succinct null checking)
- String interpolation
- nameof operator
- Dictionary initializer



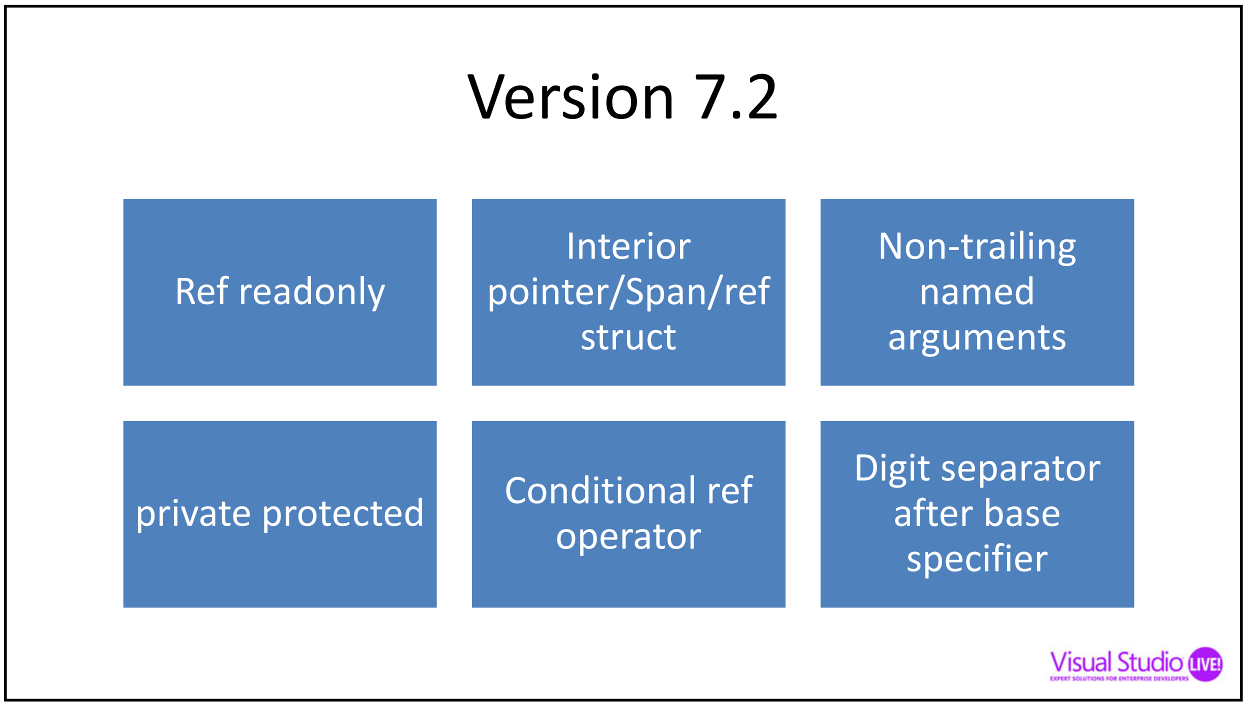
Version 7

- Out variables
- Pattern matching
- Tuples
- Deconstruction
- Discards
- Local Functions
- Binary Literals
- Digit Separators
- Ref returns and locals
- Generalized async return types
- More expression-bodied members
- Throw expressions



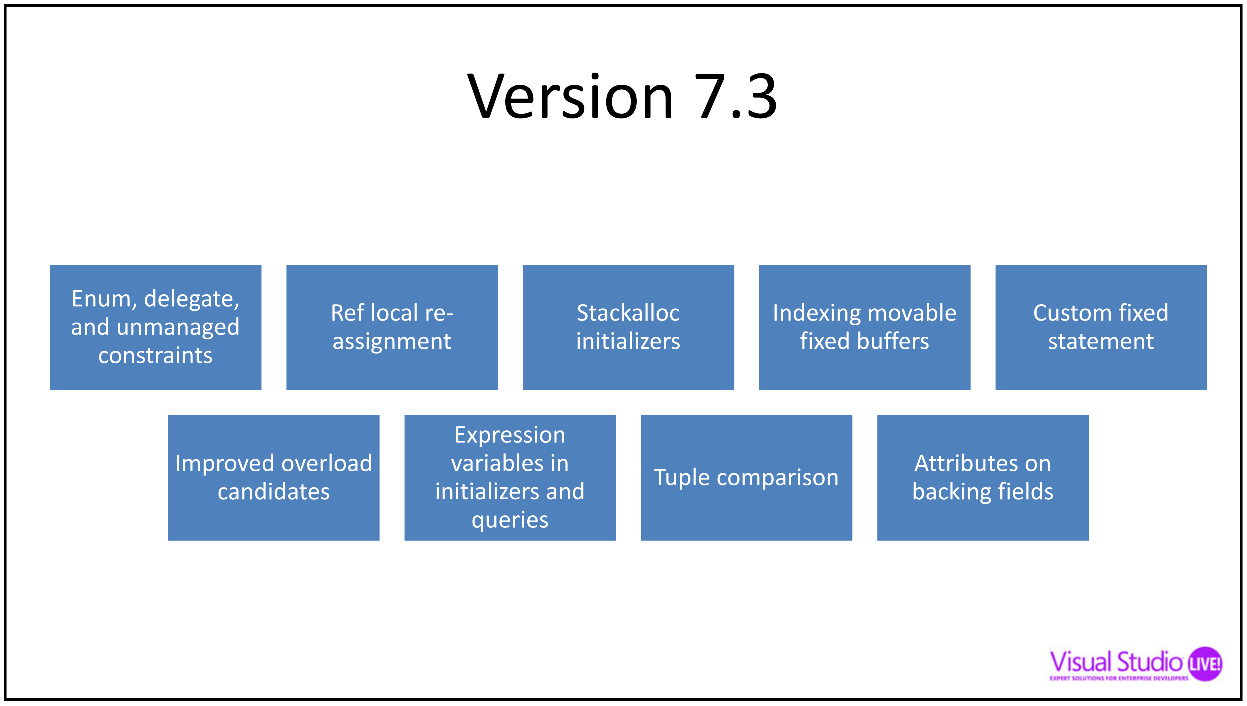
Version 7.1

- Async main
- Default expressions
- Reference assemblies
- Inferred tuple element names
- Pattern-matching with generics



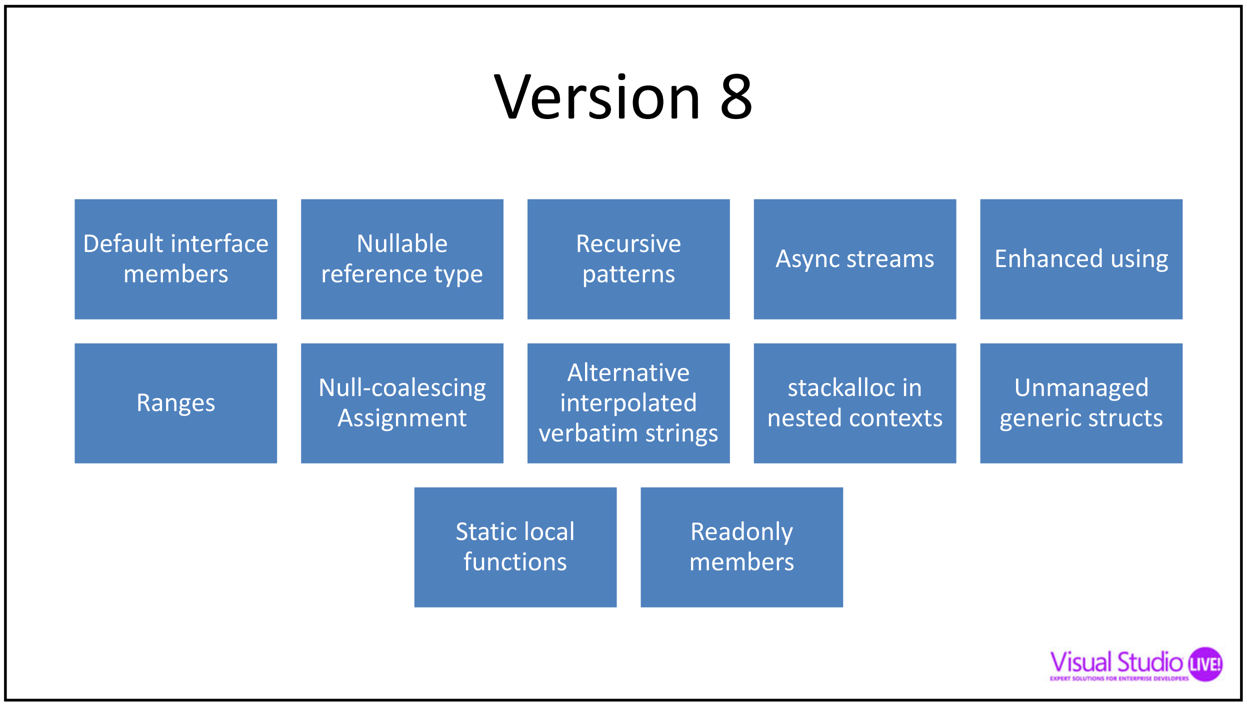
Version 7.2

- Ref readonly
- Interior pointer/Span/ref struct
- Non-trailing named arguments
- private protected
- Conditional ref operator
- Digit separator after base specifier



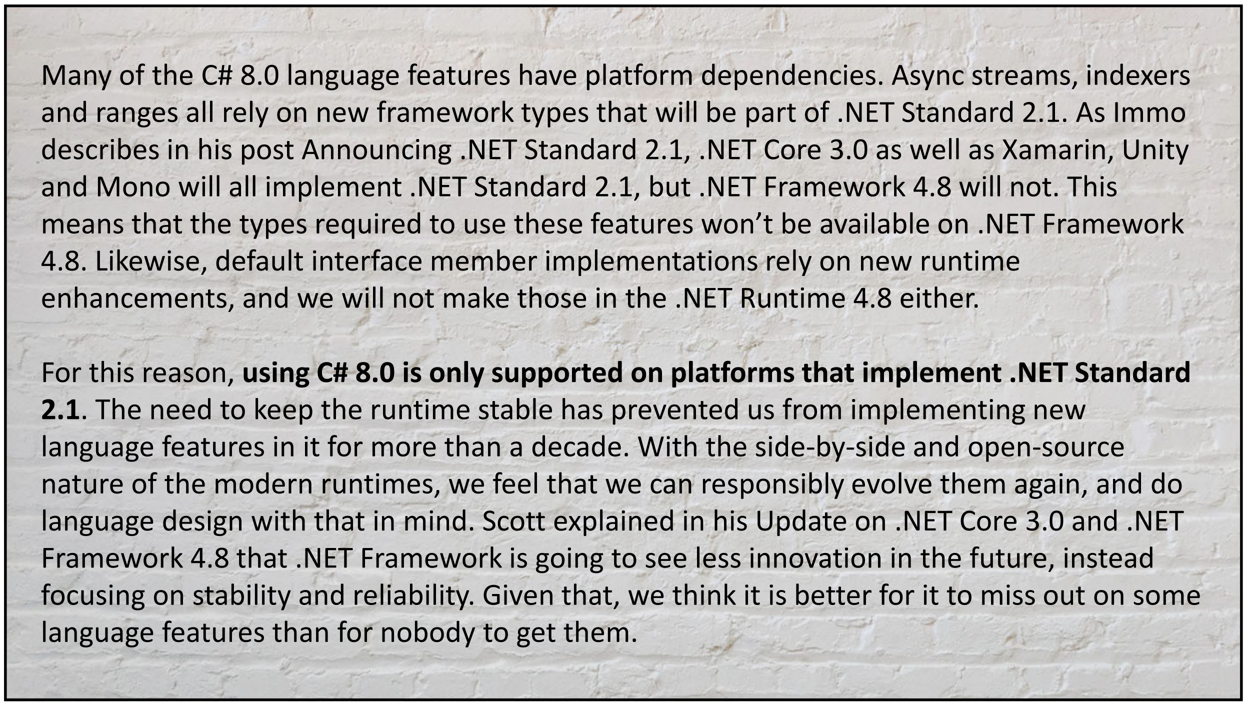
Version 7.3

- 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



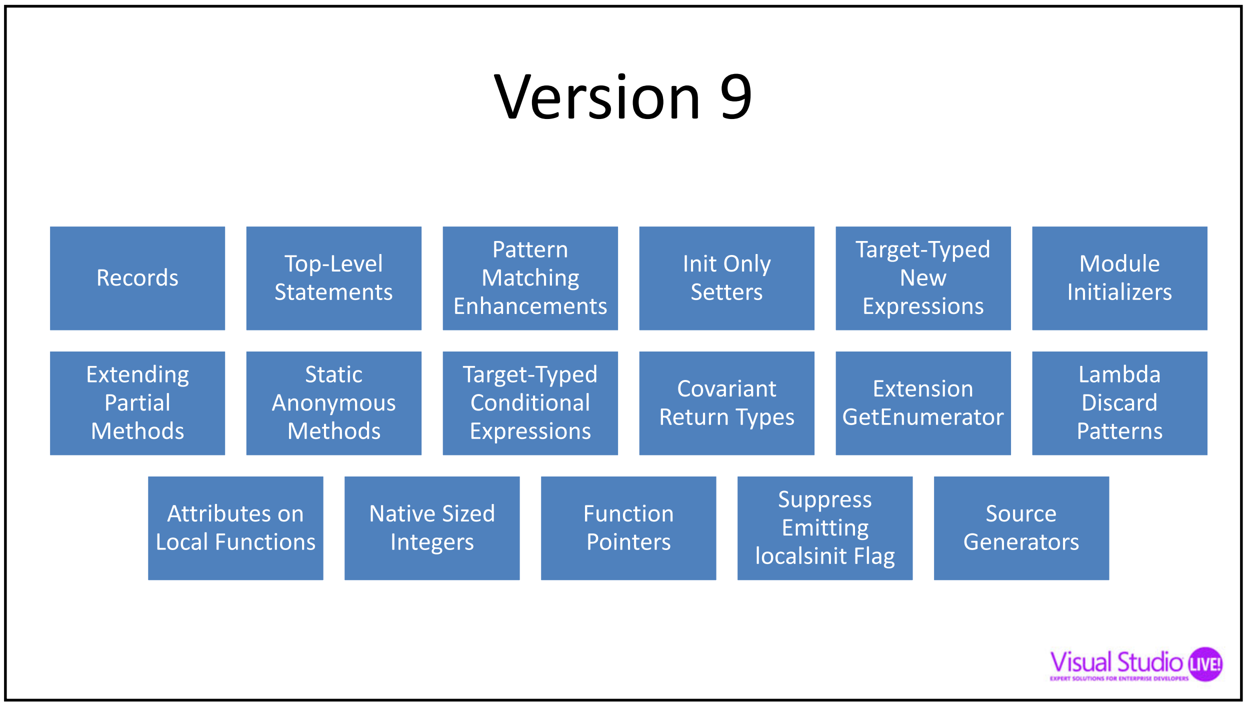
Version 8

- Default interface members
- Nullable reference type
- Recursive patterns
- Async streams
- Enhanced using
- Ranges
- Null-coalescing Assignment
- Alternative interpolated verbatim strings
- stackalloc in nested contexts
- Unmanaged generic structs
- Static local functions
- Readonly members



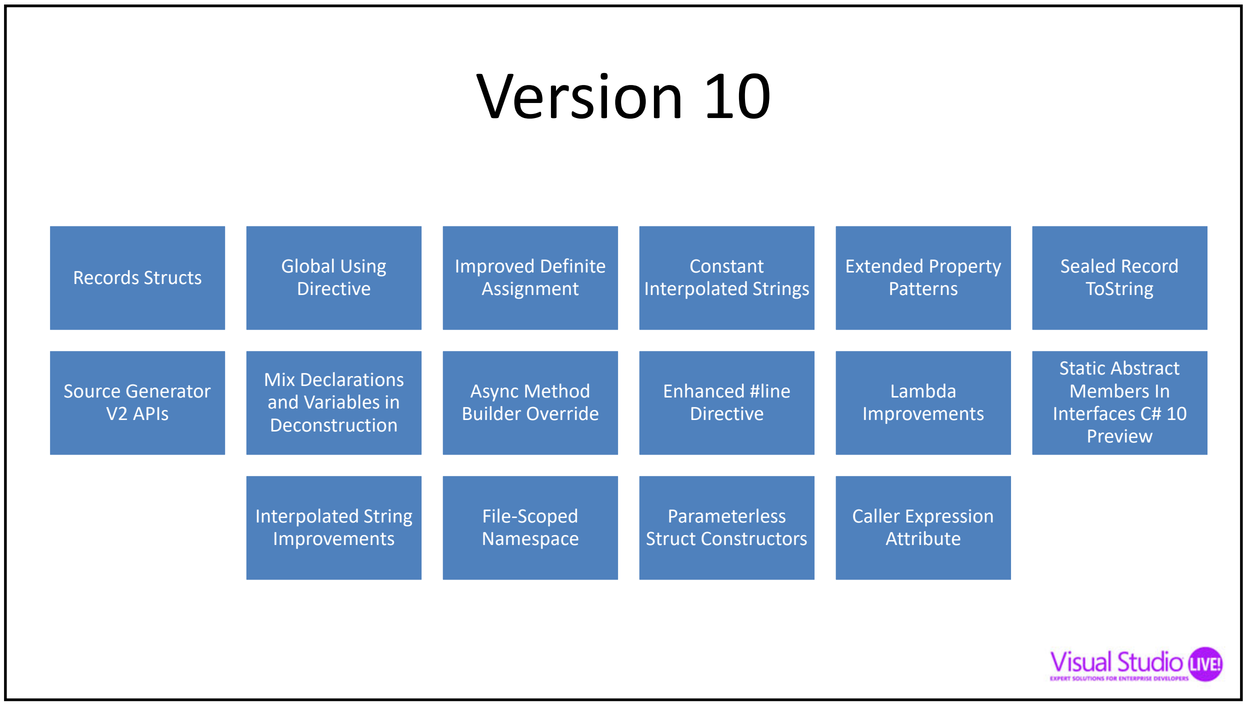
Many of the C# 8.0 language features have platform dependencies. Async streams, indexers and ranges all rely on new framework types that will be part of .NET Standard 2.1. As Immo describes in his post Announcing .NET Standard 2.1, .NET Core 3.0 as well as Xamarin, Unity and Mono will all implement .NET 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.



Version 9

- Records
- Top-Level Statements
- Pattern Matching Enhancements
- Init Only Setters
- Target-Typed New Expressions
- Module Initializers
- Extending Partial Methods
- Static Anonymous Methods
- Target-Typed Conditional Expressions
- Covariant Return Types
- Extension GetEnumerator
- Lambda Discard Patterns
- Attributes on Local Functions
- Native Sized Integers
- Function Pointers
- Suppress Emitting localsinit Flag
- Source Generators

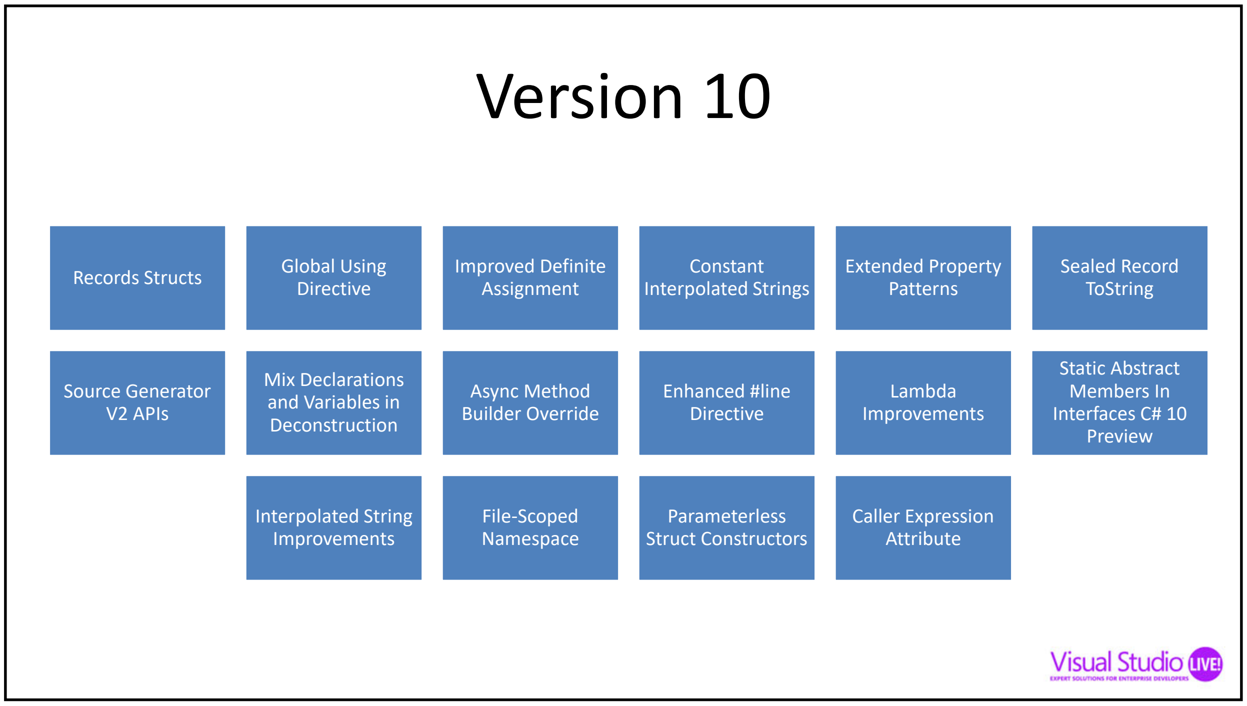


Version 10

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	

What's New in C# 10

DEMO: C# 10 FEATURES

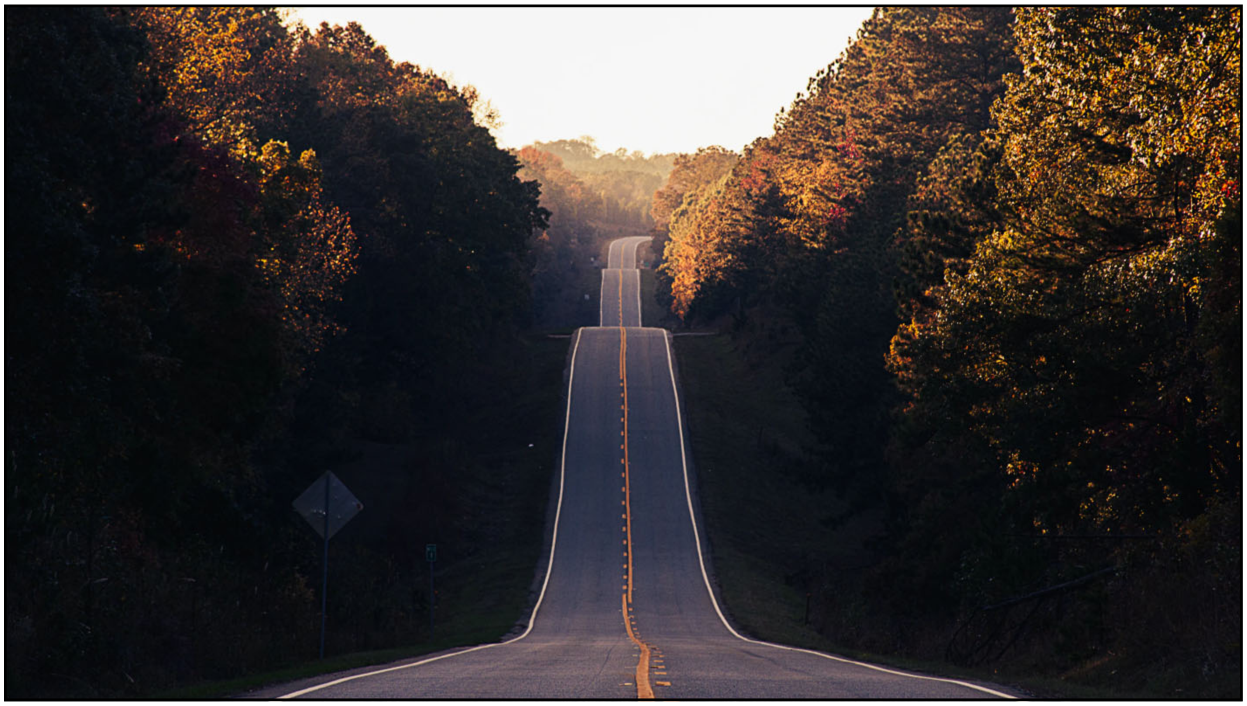


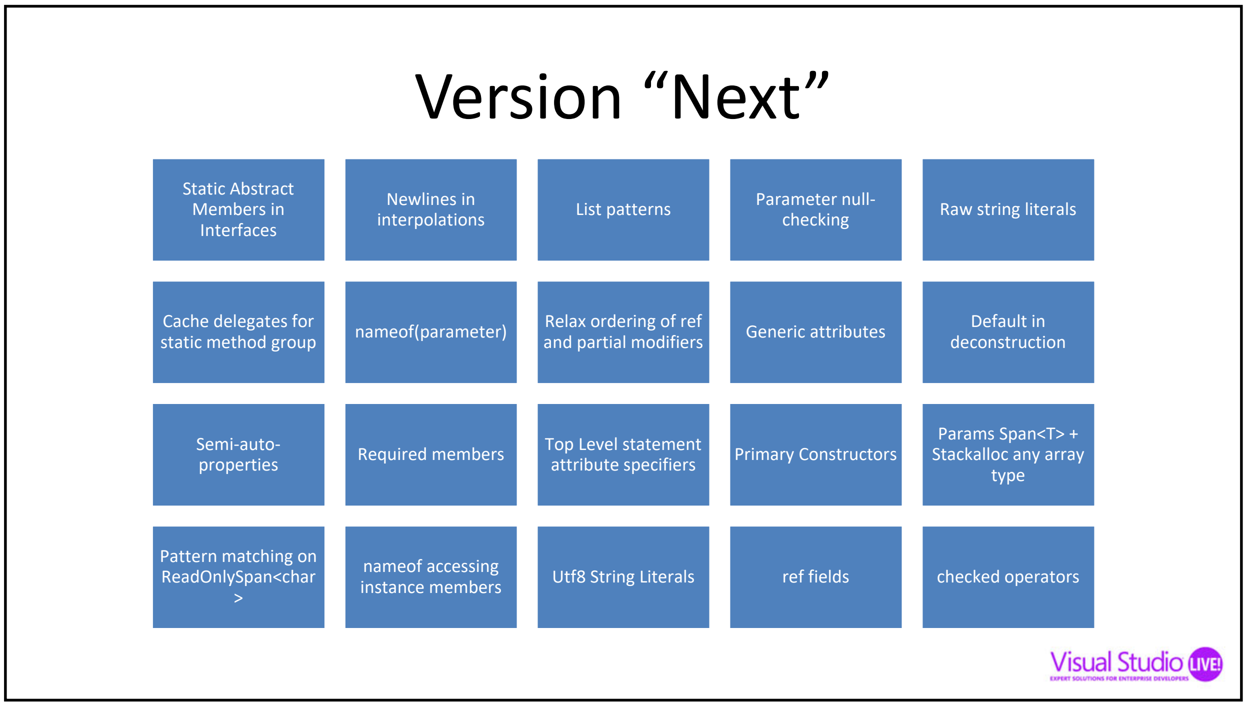
Version 10

- 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

What's New in C# 10

FUTURE DIRECTIONS



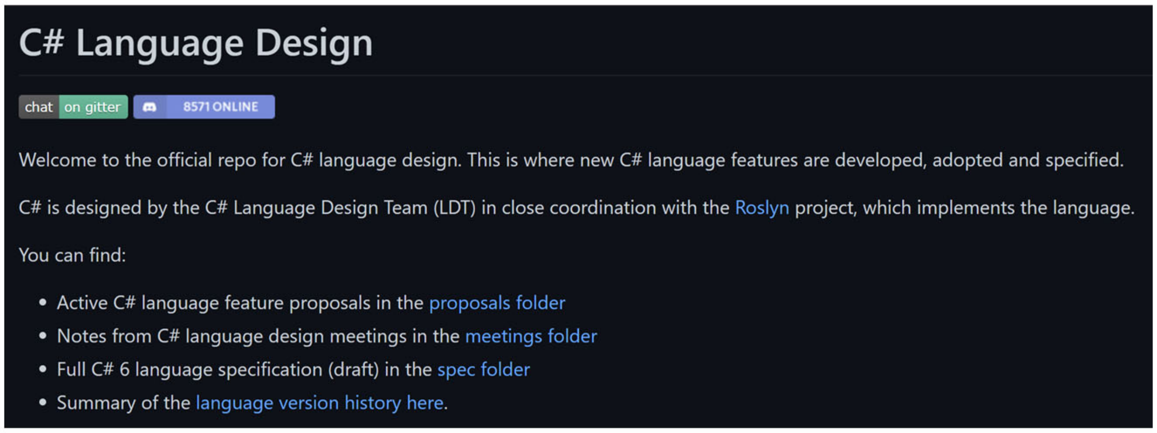


Version "Next"

Static Abstract Members in Interfaces	Newlines in interpolations	List patterns	Parameter null-checking	Raw string literals
Cache delegates for static method group	nameof(parameter)	Relax ordering of ref and partial modifiers	Generic attributes	Default in deconstruction
Semi-auto-properties	Required members	Top Level statement attribute specifiers	Primary Constructors	Params Span<T> + Stackalloc any array type
Pattern matching on ReadOnlySpan<char>	nameof accessing instance members	Utf8 String Literals	ref fields	checked operators

What's New in C# 10

DEMO: C# "NEXT" FEATURES



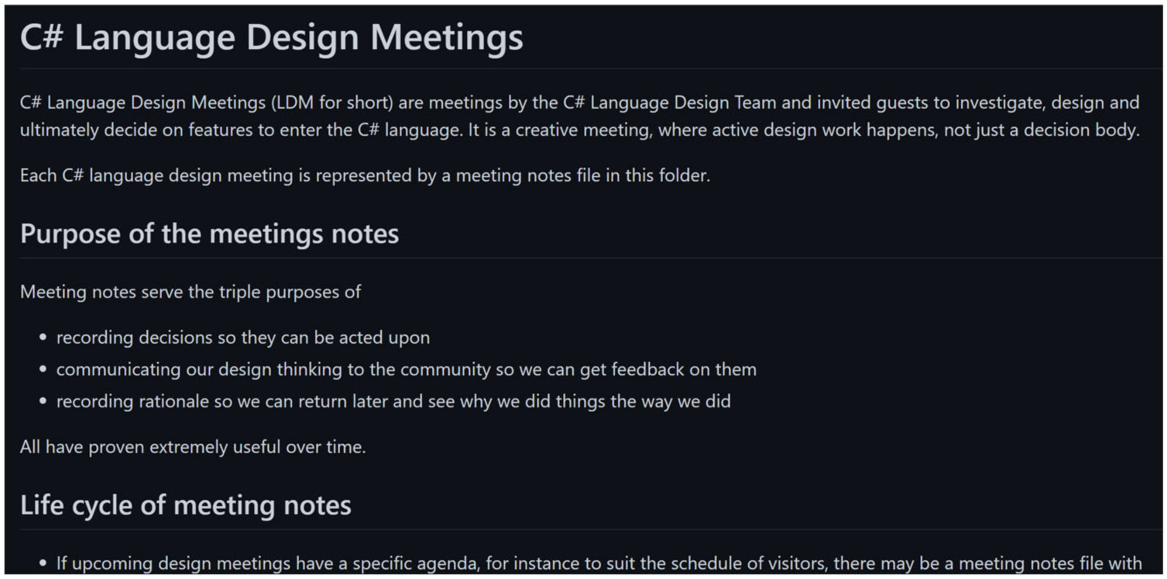
C# Language Design

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.



C# Language Design Meetings

C# Language Design Meetings (LDM for short) are meetings by the C# Language Design Team and invited guests to investigate, design and ultimately decide on features to enter the C# language. It is a creative meeting, where active design work happens, not just a decision body.

Each C# language design meeting is represented by a meeting notes file in this folder.

Purpose of the meetings notes

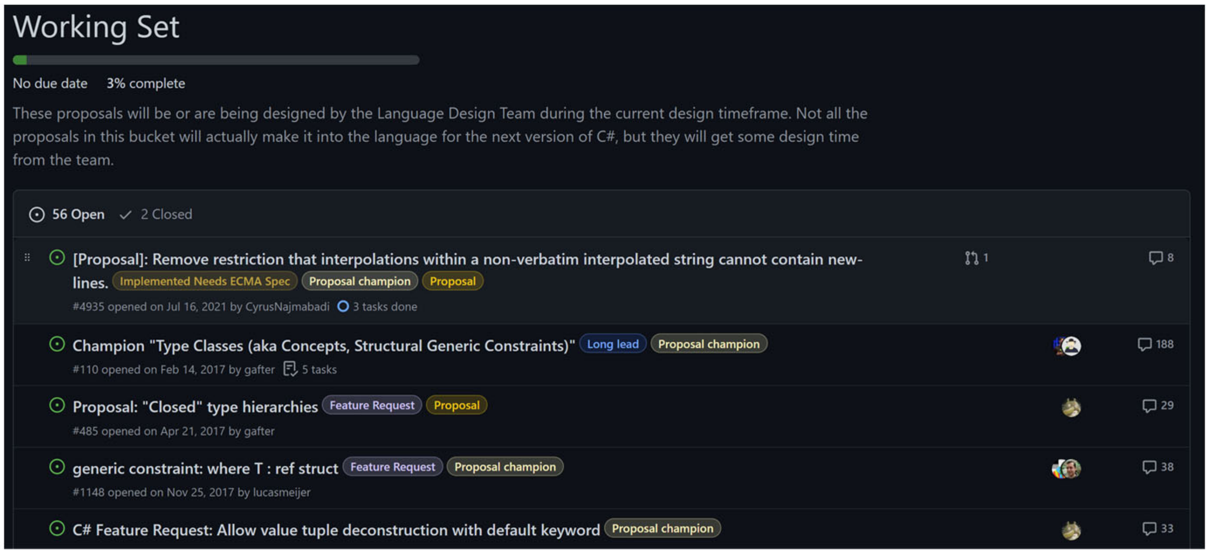
Meeting notes serve the triple purposes of

- recording decisions so they can be acted upon
- communicating our design thinking to the community so we can get feedback on them
- recording rationale so we can return later and see why we did things the way we did

All have proven extremely useful over time.

Life cycle of meeting notes

- If upcoming design meetings have a specific agenda, for instance to suit the schedule of visitors, there may be a meeting notes file with



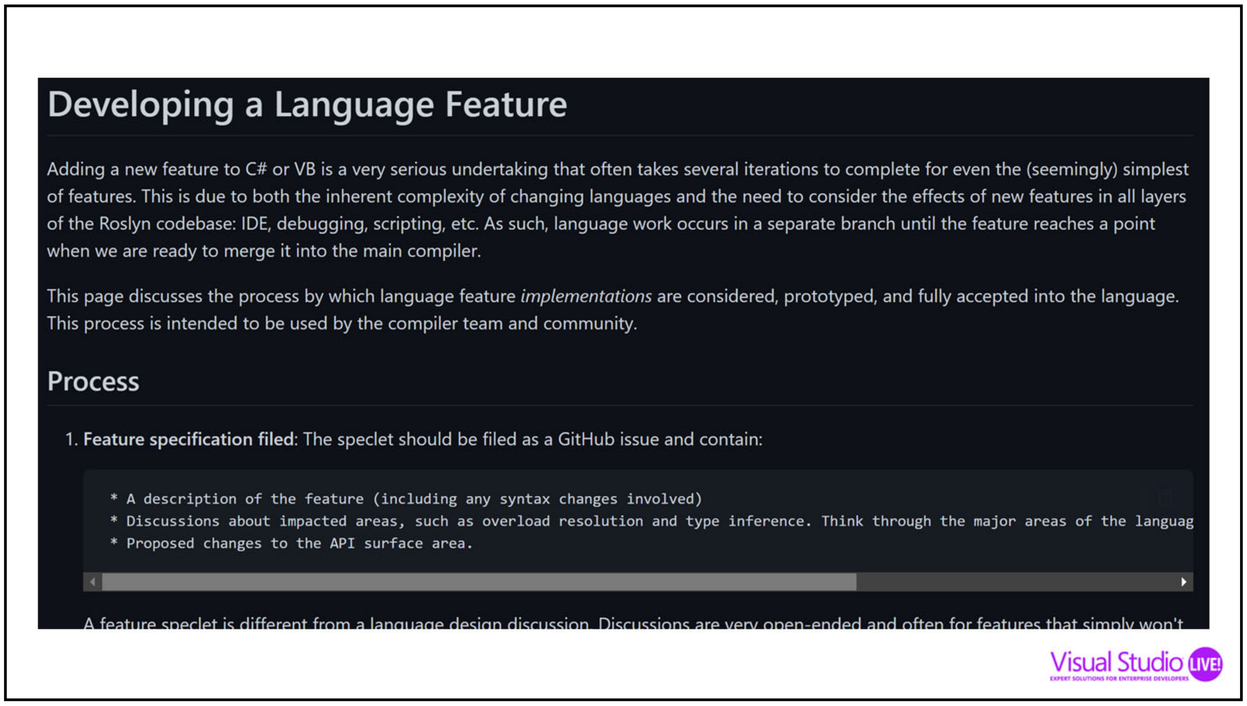
Working Set

No due date 3% 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.

56 Open ✓ 2 Closed

- [Proposal]: Remove restriction that interpolations within a non-verbatim interpolated string cannot contain new-lines. Implemented Needs ECMA Spec · Proposal champion · Proposal — 1 — 8
  #4935 opened on Jul 16, 2021 by CyrusNajmabadi 3 tasks done
- Champion "Type Classes (aka Concepts, Structural Generic Constraints)" Long lead · Proposal champion — 188
  #110 opened on Feb 14, 2017 by gafter 5 tasks
- Proposal: "Closed" type hierarchies Feature Request · Proposal — 29
  #485 opened on Apr 21, 2017 by gafter
- generic constraint: where T : ref struct Feature Request · Proposal champion — 38
  #1148 opened on Nov 25, 2017 by lucasmeijer
- C# Feature Request: Allow value tuple deconstruction with default keyword Proposal champion — 33



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 languag
* 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

Visual Studio LIVE! | AUSTIN
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

What's New in C# 10

Jason Bock
Developer Advocate
Rocket Mortgage

Level: Intermediate

#VSLIVE

NO CODE LIMITS