C# 8 & Beyond

Filip Ekberg

@fekberg


Using Statements for Static Members
Implicitly typed local variables
Extension methods
Getter & Setter separate accessibility
Null-conditional operators
Exception Filters
Anonymous methods
Named and optional parameters
Lambdas
Static classes
Auto-properties
stackalloc initializers
String interpolation
Caller info attributes
Nullable types
Digit Separators
private protected
stackalloc with Span
Dynamic binding
Default Expressions
digit separators
non-trailing named arguments
out improvements
Generics
Expression trees
Dictionary Initializers
Local Functions
Tuples and Deconstruction
Ref Assemblies
Async & Await
Infer Tuple Names
ref returns
Await inside Catch & Finally blocks
Query expressions
Iterators
nameof operator
Partial types
Improved generic constraints
Generic co- and contravariance
Pattern Matching
Object and collection initializers
conditional ref operator
Async Main
Tuple equality
Binary Literals
Expression-bodied members
Auto-Property Initializers
Anonymous types
ref readonly & in parameter

@fekberg

jcouv Update Language Feature Status (#38818) 0056d76 15 days ago

13 contributors

98 lines (79 sloc) 14.6 KB Raw Blame History

Language Feature Status

This document reflects the status, and planned work in progress, for the compiler team. It is a live document and will be updated as work progresses, features are added / removed, and as work on feature progresses. This is not an exhaustive list of our features but rather the ones which have active development efforts behind them.

C# Next

Feature	Branch	State	Developers	Reviewer	LDM Champ
Caller expression attribute	caller-expression	Prototype	alrz	jcouv	jcouv
Target-typed new	target-typed-new	Prototype	alrz	jcouv	jcouv
Generic attributes	generic-attributes	In Progress	AviAvni	agocke	mattwar
Default in deconstruction	decon-default	Implemented	jcouv	gafter	jcouv
Relax ordering of `ref` and `partial` modifiers	ref-partial	In Progress	alrz	gafter	jcouv
Parameter null-checking	param-nullchecking	In Progress	fayrose	agocke	jaredpar

dotnet / csharplang

Watch 618 | Star 4,829 | Fork 539

Code | Issues 1,705 | Pull requests 29 | Projects 4 | Wiki | Security | Insights

Labels | Milestones | New issue

8.0

Due by January 01, 2080 11% complete

Candidates for the C# 8.0 release

23 Open ✓ 3 Closed

- Issues in Default Interface Methods [Spec] — 141
  #406 opened on Apr 4, 2017 by gafter
- Permit lambda and local function parameters to hide names from the enclosing scope (16.3, Core 3) [Proposal champion] — 15
  #2777 opened on Sep 4 by gafter
- Champion "permit `t is null` for unconstrained type parameter" (16.3, Core 3) [Proposal champion] [Smallish Feature] — 19
  #1284 opened on Jan 26, 2018 by gafter 3 of 5
- Champion: Target-typed switch expression (16.3, Core 3) [Proposal] [Proposal champion] — 33
  #2389 opened on Apr 4 by gafter 3 of 5
- Champion "Null-coalescing assignments" (16.3, Core 3) [Proposal champion] [Smallish Feature] — 99
  #34 opened on Feb 9, 2017 by MadsTorgersen 2 of 5
- Champion: "Allow Obsolete attribute on getters and setters" (16.3, Core 3) [Proposal champion] — 5
  #2152 opened on Jan 17 by gafter 3 of 5
- Open LDM Issues for Nullable Reference Types — 13
  #2201 opened on Feb 6 by cston
- Champion: extensions in pattern-based statements (16.3, Core 3) [Proposal champion] — 12
  #2135 opened on Jan 11 by jcouv 0 of 5

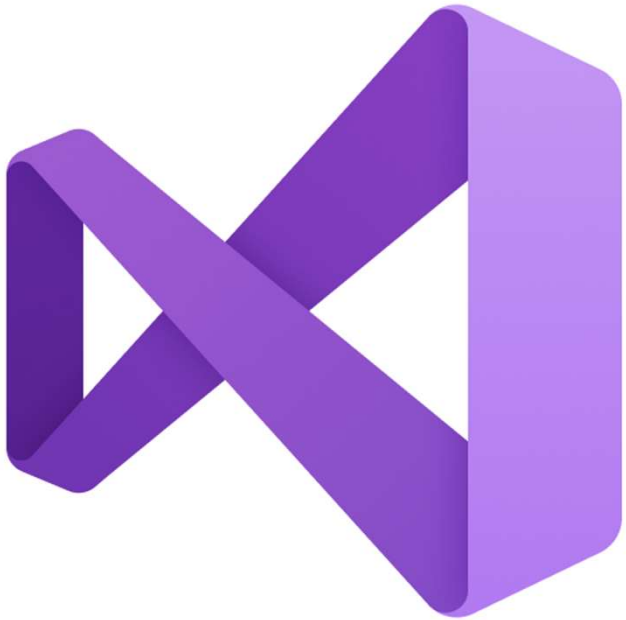
@fekberg

8.0

Due by January 01, 2080 **11%** complete

Candidates for the C# 8.0 release

@fekberg




C# 8.0

C# 8.0

Default Interface Methods

Nullable reference type

Recursive patterns

Async streams

Enhanced using

Ranges

Null-coalescing Assignment

Static local functions

Readonly members

Alternative interpolated verbatim strings

stackalloc in nested contexts

Unmanaged generic structs

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


C# X


Don’t be sad when they
don’t make the cut

Target-type new expression

```
Triangle triangle = new();

Dictionary<string, List<int>> field = new() {
    { "item1", new() { 1, 2, 3 } }
};
```

https://github.com/dotnet/csharplang/blob/master/proposals/target-typed-new.md

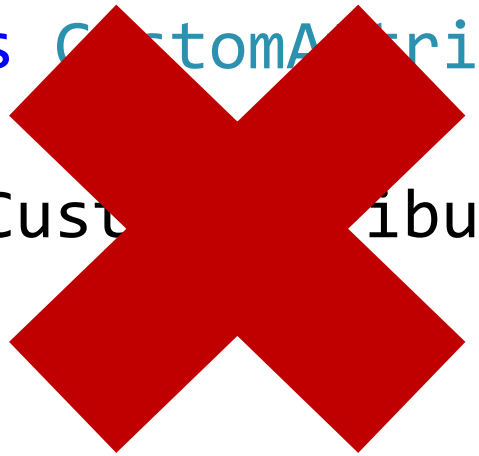
default in deconstruction

```
(int age, string name) = (default, default);
```

```
(int age, string name) = default;
```

https://github.com/dotnet/csharplang/issues/1394

Generic attributes



```
public class CustomAttribute : Attribute
{
    public CustomAttribute(Type t)
    {

    }
}
```

```
public class CustomAttribute<T> : Attribute {}
```

https://github.com/dotnet/roslyn/issues/953

Caller expression attribute

```
void Assert(bool condition,
        [CallerArgumentExpression("condition")]
        string message = null);
```

```
Debug.Assert(array != null);
Debug.Assert(array.Length == 1);
```

```
Debug.Assert(array != null, "array != null");
Debug.Assert(array.Length == 1, "array.Length == 1");
```

https://github.com/dotnet/csharplang/blob/master/proposals/caller-argument-expression.md

Negated-condition if statement

```
if( !(shape is Triangle) ) {};
```

```
if(shape is not Triangle) {};

if!(shape is Triangle) {};

unless(shape is Triangle) {};
```

https://github.com/dotnet/csharplang/issues/882

Null-conditional await

```
(task == null) ? null : await task
```

```
await? task
```

Record Types

```
public class Triangle : Shape
{
    public int A { get; }
    public int B { get; }
    public int C { get; }

    public Triangle(int a, int b, int c)
    {
        A = a;
        B = b;
        C = c;
    }
}
```

```
record Triangle(int A, int B, int C) : Shape {}
```

https://github.com/dotnet/csharplang/issues/39

Record Types

```
public class Triangle : Shape, IEquatable<Triangle>
{
    public int A { get; }
    public int B { get; }
    public int C { get; }

    public Triangle(int a, int b, int c)
    {
        A = a; B = b; C = c;
    }

    public bool Equals(Triangle other)
        => other != null && Equals(A, other.A) && Equals(B, other.B) && Equals(C, other.C);

    public override bool Equals(object other) => this.Equals(other as Triangle);

    public override int GetHashCode() =>  A.GetHashCode() * 17 + B.GetHashCode() + C.GetHashCode();

    public Triangle With(int A = this.A, int B = this.B, int C = this.C) => new Triangle(A, B, C);

    public void Deconstruct(out int A, out int B, out int C)  { A = this.A; B = this.B; C = this.C; }
}
```

Declaration expressions

```
char ch;
while ((ch = GetNextChar()) == 'a'
        || ch == 'b'
        || ch == 'c')
{ }
```

```
while ((char ch = GetNextChar()) == 'a'
            || ch == 'b'
            || ch == 'c')
{ }
```

https://github.com/dotnet/csharplang/issues/973

Dictionary literals

```
var x = new Dictionary<string, int>() {
    { "foo", 4 },
    { "bar", 5 }
};

var x = ["foo": 4, "bar": 5];
```

Extension Everything

```
interface IFoo<T>
{
    public T Bar { get; }
}

extension IntFoo of int : IFoo<int>
{
    public int Bar => 100;
}

int x = 10;

var y = x.Bar;
```

https://github.com/dotnet/roslyn/issues/11159

As well as..

Type classes

Lambda discard parameters

Parameter null-checking

Relax ordering of ref and partial modifiers

Native ints

params Span<T>

Attributes on local functions

Function pointers

&&= and ||= assignment operators

@fekberg

Ranges

Async Streams & Enumerables

Record Types

Target-type new expression

default in deconstruction

Generic attributes

Caller expression attribute

Enhanced using (pattern-based using)

Default interface methods

Nullable reference type

Null coalescing assignment

Switch expressions

Declaration expressions

Dictionary literals

Type classes

&&= and ||= assignment operators

static local functions

params Span<T>

readonly struct members

Negated-condition if statement

Null-conditional await

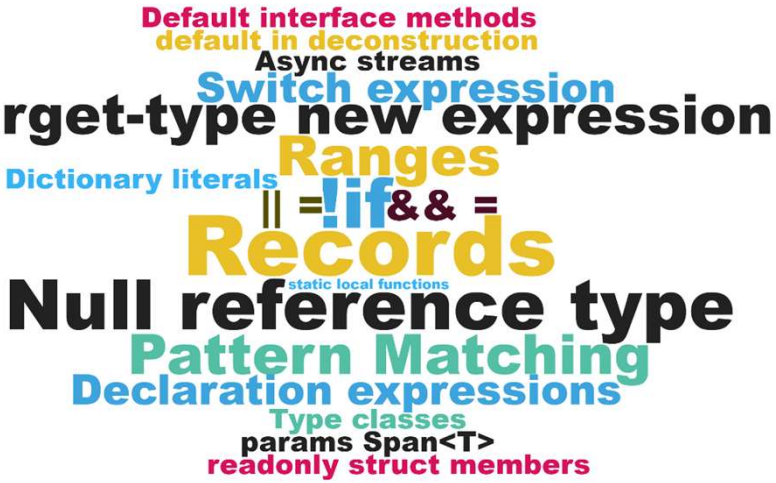
And More!

@fekberg

C# 1 => 7.3

+

C# 8 => n



@fekberg

Thanks!

I'm Filip Ekberg

@fekberg

C# smorgasbord free @ filipekberg.se

PLURALSIGHT

fekberg