Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

ROCK YOUR CODE TOUR · 2017 | CHICAGO

What's New in C#7

Jason Bock
Practice Lead,
Magenic

Personal Info

- http://www.magenic.com
- http://www.jasonbock.net
- https://www.twitter.com/jasonbock
- https://www.github.com/jasonbock
- jasonb@magenic.com

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Downloads

https://github.com/JasonBock/WhatsNewInCSharp7

https://github.com/JasonBock/WhatsNewInCSharp6

https://www.slideshare.net/JasonBock2/whats-new-in-c7-79030221

Overview

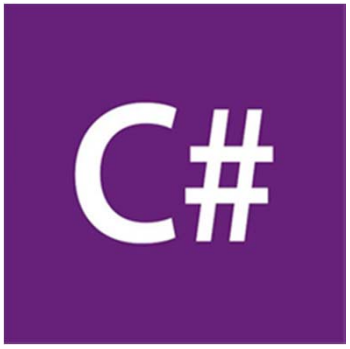
- Language Evolution
- C#7 and C#7.1 Features
- Future Directions

Remember…
https://github.com/JasonBock/WhatsNewInCSharp7

https://github.com/JasonBock/WhatsNewInCSharp6

https://www.slideshare.net/JasonBock2/whats-new-in-c7-79030221

Language Evolution



Est. 2002

http://wilsonwong.me/images/logos/csharplogo.png

Language Evolution

Version 1

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

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

Language Evolution

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

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

Language Evolution

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

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

Language Evolution

Version 4

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

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

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

Language Evolution

Version 5

- Asynchronous methods
- Caller info attributes

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

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

Language Evolution

http://static.hdw.eweb4.com/media/wallpapers_1920x1080/fantasy/1/1/dark-tower-fantasy-hd-wallpaper-1920x1080-5771.jpg

Language Evolution

The .NET Foundation is an independent organization to foster open development and collaboration around the growing collection of open source technologies for .NET

http://www.dotnetfoundation.org/

Language Evolution

README.md

Welcome to the .NET Compiler Platform ("Roslyn")

Windows - Unit Tests

Branch	Debug x86	Debug x64	Release x86	Release x64	Determinism	Debug Integration	Release Integration
master	build passing	build passing	build passing	build passing	build passing	build passing	build passing
dev15.0.x	build passing	build passing	build passing	build passing	build passing	build passing	build passing
dev15.3.x	build passing	build passing	build passing	build passing	build passing	build passing	build passing
dev16	build passing	build passing	build passing	build passing	build passing	build passing	build passing

Linux/Mac - Unit Tests

Branch	Ubuntu14	Ubuntu16	macOS
master	build passing	build passing	build passing
dev15.0.x	build passing		build passing
dev15.3.x	build passing	build passing	build passing
dev16	build passing	build passing	build passing

gitter join chat

Roslyn provides open-source C# and Visual Basic compilers with rich code analysis APIs. It enables building code analysis tools with the same APIs that are used by Visual Studio.

https://github.com/dotnet/roslyn

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

Language Evolution

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

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

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

Language Evolution

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

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

Language Evolution

Version 7.1

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

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

What's New in C#7

DEMO: C#7 AND C#7.1 FEATURES

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

C#7 Features

- Binary Literals and Digit Separators
  - Declaring constants and values using "spacers"
    - `const int FirstValue = 00_11_00_11;`
  - You have to put "0b" in front of the variable though
    - `const int SecondValue = 0b00_11_00_11;`
  - Spacers also work with other declarations
    - `const int ThirdValue = 0x3_3;`

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

C#7 Features

- Local functions
  - If you have processing that is reused in a function but only in that function, declare a local function
  - Local functions can “see” variables in the function
  - Declaration of the local function can be anywhere within the function

```
private static void UseLocalFunction()
{
  uint Collatz(uint value) =>
    value % 2 == 1 ? (3 * value + 1) / 2 : value / 2;
```

C#7 Features

- Out variables
  - Typically with out variables you need to declare the variable first, then pass it into the method
  - Now you can do that all in one line

```
if (int.TryParse(value, out var thirdWay))
```

C#7 Features

- Tuples and Deconstruction
  - You need to reference `System.ValueTuple` for this to work
  - This makes it easy to return multiple values
  - You can also turn an object into a tuple if it implements `Deconstruct()`

```
(var x, var y) = result;
Console.Out.WriteLine($"{nam
```

```
Console.Out.WriteLine($"{nameof(Calculator.CalculateWithDeconstructedType
(var x, var y) = result;
Install package 'System.ValueTuple'   Find and install latest version
                                      Install with package manager...
var tupleResult = Calculator.Ca
    Program.FirstValue, Program.
```

CS8179 Predefined type 'System.ValueTuple`2' is not defined or imported

Find and install latest version of 'System.ValueTuple'

C#7 Features

- Pattern matching
  - Pattern matching "are syntactic elements that can test that a value has a certain 'shape', and extract information from the value when it does." - https://blogs.msdn.microsoft.com/dotnet/2017/03/09/new-features-in-c-7-0/
  - It's not as full-featured as pattern matching in languages like F# and Rust, but it's a start

```
private static void CheckThings(Thing thing)
{
  switch (thing)
  {
    case SomeThing someThing22 when someThing22.Value == 22:
      Console.Out.WriteLine(
        $"{nameof(SomeThing)} with {nameof(Thing.Value)} == 22");
      break;
```

C#7 Features

- Ref returns and locals
  - If you return a value type from a method, the value is copied
  - For perf reasons, you may want to have “by reference” semantics
  - With ref returns and ref locals, you can now do that.

```
ref var fastStruct1 = ref fastList[50];
ref var fastStruct2 = ref fastList[50];
fastStruct1.Value1 = 22;
Console.Out.WriteLine(
  $"{nameof(fastStruct2)}.{nameof(fastStruct2.Value1)} is {fastStruct2.Value1}");
```

C#7 Features

- Generalized async return types
  - Typically async methods return Task or Task<T>
  - Now you can return “Task-like” tasks objects
  - One example on how to do this is ValueTask<T>: https://github.com/dotnet/corefx/blob/5a6d8ca975b512eeeea7404c740afcf865128405/src/System.Threading.Tasks.Extensions/src/System/Threading/Tasks/ValueTask.cs

```
private static void ShowValueTask()
{
  ValueTask<int> ReturnValueAsync() =>
    new ValueTask<int>(22);
```

C#7 Features

- More expression-bodied members and throws expressions
  - You can now do expression-bodied members with constructors, finalizers and accessors
  - You can also throw exceptions from expressions, which makes the “if parameter is `null` throw `ArgumentNullException`” pattern easier to write.

```
public sealed class ExpressedPerson
  : INotifyPropertyChanged
{
  public event PropertyChangedEventHandler PropertyChanged;

  private int id;
  private string name;

  public ExpressedPerson(int id) => this.id = id;

  public ExpressedPerson(int id, string name)
    : this(id) =>
      this.name = name ?? throw new ArgumentNullException(nameof(name));
```

C#7.1 Features

- Async Main
  - Asynchronous methods cause the entry point to be `Main()`, but that wasn’t allowed. Now you can write that!
  - Note that the `SynchronizationContext` is not set

```
class Program
{
  static async Task Main(string[] args) =>
    await Program.DoValueTaskAsync();

  private static async Task DoValueTaskAsync()
  {
    ValueTask<int> ReturnValueAsync() =>
      new ValueTask<int>(22);

    Console.Out.WriteLine(await ReturnValueAsync());
    Console.Out.WriteLine(SynchronizationContext.Current == null);
  }
```

C#7.1 Features

- Default expressions
  - You could set variables and arguments to their default value using `default()`
  - Now you can just use the `default` keyword by itself

```
private static void ShowDefaultLiterals()
{
  int DoSomething(int x, string y) =>
    x == default && y == default ? default : 22;

  Console.Out.WriteLine(DoSomething(default, "data"));
  Console.Out.WriteLine(DoSomething(22, default));
  Console.Out.WriteLine(DoSomething(default, default));
  Console.Out.WriteLine(DoSomething(0, null));
```

C#7.1 Features

- Inferred tuple element names
  - Before 7.1, assigning a variable tuple values did not infer names.
  - Now it does!

```
private static void ShowInferredTupleNames()
{
  var x = (id: 3, name: "name");
  var y = (x.id, x.name);
  Console.Out.WriteLine($"{y.id}, {y.name}");
}
```

C#7.1 Features

- Pattern matching with generics
  - There was an issue in 7.0 where generics couldn't be used in pattern-matching
  - In 7.1 this has been relaxed/fixed

```
private static void ShowPatternMatchingWithGenerics()
{
  void HandleThing<T>(T thing) where T : Thing
  {
    if(thing is SomeThing someThing)
    {
      Console.Out.WriteLine($"It's {someThing.GetType().Name}");
    }
    else if (thing is AnotherThing anotherThing)
    {
      Console.Out.WriteLine($"It's {anotherThing.GetType().Name}");
    }
    else
    {
      Console.Out.WriteLine($"Unknown {nameof(Thing)}");
    }
  }

  HandleThing(new SomeThing());
  HandleThing(new AnotherThing());
  HandleThing(new SneakyThing());
}
```

Future Directions

https://dialectline.files.wordpress.com/2012/04/crystal_ball1.jpg

Future Directions

C# 7.2

Feature	Branch	State	Developers	Reviewer	LDM Champ
ref readonly	readonly-ref	Prototype	vsadov, omar	cston	jaredpar
blittable	None	Proposal	None		jaredpar
strongname	strongname	In Progress	Ty Overby		jaredpar
interior pointer	None	Proposal	vsadov	jaredpar	jaredpar
non-trailing named arguments	non-trailing	Prototype	jcouv	TBD	jcouv

C# 8.0

Feature	Branch	State	Developers	Reviewer	LDM Champ
Default Interface Methods	defaultInterfaceImplementation	Prototype	AlekseyTs	gafter	gafter
Nullable reference type	NullableReferenceTypes	Prototype	cston, AlekseyTs		mattwar

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

C#7.2 – Ref Readonly

```
static Vector3 Add(ref readonly Vector3 v1, ref readonly Vector3 v2)
{
  // not OK!!
  v1 = default(Vector3);

  // not OK!!
  v1.X = 0;

  // not OK!!
  foo(ref v1.X);

  // OK
  return new Vector3(v1.X + v2.X, v1.Y + v2.Y, v1.Z + v2.Z);
}
```

```
static Vector3 Add(in Vector3 v1, in Vector3 v2)
{
  // OK
  return new Vector3(v1.X + v2.X, v1.Y + v2.Y, v1.Z + v2.Z);
}
```

C#7.2 – Blittable

```
blittable struct Point
{
  public int X;
  public int Y;
}
```

C#7.2 – Non-trailing named arguments

```
public void DoSomething(bool isEmployed, string personName, int personAge) { ... }

DoSomething(isEmployed: true, name, age); // currently CS1738, but would become legal
DoSomething(true, personName: name, age); // currently CS1738, but would become legal
DoSomething(name, isEmployed: true, age); // remains illegal
DoSomething(name, age, isEmployed: true); // remains illegal
DoSomething(true, personAge: age, personName: name); // already legal
```

C#8 – Default Interface Methods

```
interface IA
{
  void M() { WriteLine("IA.M"); }
}

class C : IA { } // OK

IA i = new C();
i.M(); // prints "IA.M"
```

C#8 – Nullable Reference Types

```
string? n; // Nullable reference type
string s;  // Non-nullable reference type

n = null; // Sure, it's nullable
s = null; // Warning! Shouldn't be null!
s = n;    // Warning! Really!

WriteLine(s.Length); // Sure; it's not null
WriteLine(n.Length); // Warning! Could be null!

if(n != null) { WriteLine(n.Length); } // Sure, you checked.
WriteLine(n!.Length);                  // OK, if you insist!
```

Future Directions

- Source generation (metaprogramming!)
- private protected
- Records and With expressions
- Shapes
- Generic attributes

Future Directions

```
public shape SGroup<T>
{
  static T operator +(T t1, T t2);
  static T Zero { get; }
}

public extension IntGroup of int : SGroup<int>
{
  public static int Zero => 0;
}

public static AddAll<T>(T[] ts)
  where T : SGroup<T>
{
  var result = T.Zero;
  foreach (var t in ts) { result += t; }
  return result;
}

int[] numbers = { 5, 1, 9, 2, 3, 10, 8, 4, 7, 6 };
WriteLine(AddAll(numbers));
```

Future Directions

dotnet / csharplang

Unwatch 284 | Star 1,306 | Fork 131

Code | Issues 592 | Pull requests 16 | Projects 0 | Insights

Filters | is:open label:"Feature Request" | Labels | Milestones | New issue

Clear current search query, filters, and sorts

364 Open | 30 Closed | Author | Labels | Projects | Milestones | Assignee | Sort

- Proposal: Enable Downcasting Objects Discussion Feature Request — #830 opened a day ago by doktrova — 9
- Proposal: extend tuple projection initializers to preserving element names through unnamed tuples Discussion Feature Request — #829 opened 2 days ago by DavidArno — 9
- [Proposal] null guards for nullable types Discussion Feature Request — #828 opened 2 days ago by mariusGundersen — 8
- Please Consider to Simplify the Keyword of "await"! Discussion Feature Request — #827 opened 3 days ago by juepiezhongren — 0 of 2 — 66
- Let value types participate in contravariance Discussion Feature Request — #826 opened 3 days ago by Porges — 7

https://github.com/dotnet/csharplang/labels/Feature%20Request

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

Future Directions

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:

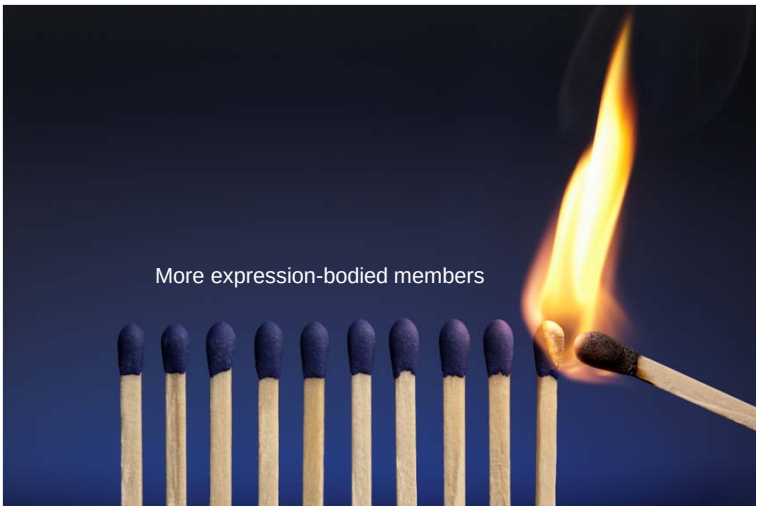
```
* A description of the feature (including any syntax changes involved)
* Discussions about impacted areas, such as overload resolution and type inference. Think through the major are
* Proposed changes to the API surface area.
```

A feature speclet is different from a language design discussion. Discussions are very open-ended and often for features

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

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

Future Directions



http://www.wildfiresparks.co.uk/wp-content/uploads/2014/02/Influence.png

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

ROCK YOUR CODE TOUR 2017 | CHICAGO

What's New in C#7

Jason Bock
Practice Lead, Magenic

Remember…
- https://github.com/JasonBock/WhatsNewInCSharp7
- https://github.com/JasonBock/WhatsNewInCSharp6
- https://www.slideshare.net/JasonBock2/whats-new-in-c7-79030221
- References in the notes on this slide