

C# History (before Roslyn)

2002	• C# 1/VB7
2005	• C# 2: Generics, Iterators, Anonymous Delegates, Nullable, Partial, co/contravariance
2008	• C# 3: LINQ, Lambda, Anonymous, AutoProp, Extension Methods, Expression
2010	• C# 4: Parallel, Dynamic, optional params, named arguments, embedded interop
2012	• C# 5: Async, Caller info
2013	• ???
2015	• Roslyn



C# Post Roslyn

2015	• C# 6: Static imports, prop initializers, null propagator, string interpolation, nameof
12/2016	• C# 7.0: Out variables, tuples, pattern matching, local functions, ref locals and returns
8/2017	• C# 7.1: Async main, default literal, inferred tuple names, reference assembly gen
8/2017	• C# 7.2: ref semantics with value types, non-trailing named args, numeric literals, private protected
??	• C# 7.3: tuple equality, constraints, expression variables, custom fixed, ref reassignment
???	• C# 8.0: Default interface, nullable reference types, async streams, generic attributes, ranges

<https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/>
<https://github.com/dotnet/roslyn/blob/master/docs/Language%20Feature%20Status.md>



Named Tuples

Binary Literals

Digit Separators

Local Functions

Pattern Matching

```
static void Main(string[] args)
{
    object[] numbers = { 0b1, 0b10, new object[] { 0b10, 0b100, 0b1000 }, 0b1000_0, 0b1000_00 };
    (int sum, int count) Tally(IEnumerable<object> list)
    {
        var r = (sum:0, count:0);
        foreach (var v in list)
        {
            switch(v)
            {
                case int i:
                    r.sum += i, r.count++;
                    break;
                case IEnumerable<object> l when l.Any():
                    var t1 = Tally(l);
                    r.s += t1.sum; r.c += t1.count;
                    break;
                case null:
                    break;
            }
        }
        return r;
    }
    var result = Tally(numbers);
    Console.WriteLine($"Sum: {result.sum} Count: {result.count}");
}
```

C#7

C# Scripting / Interactive

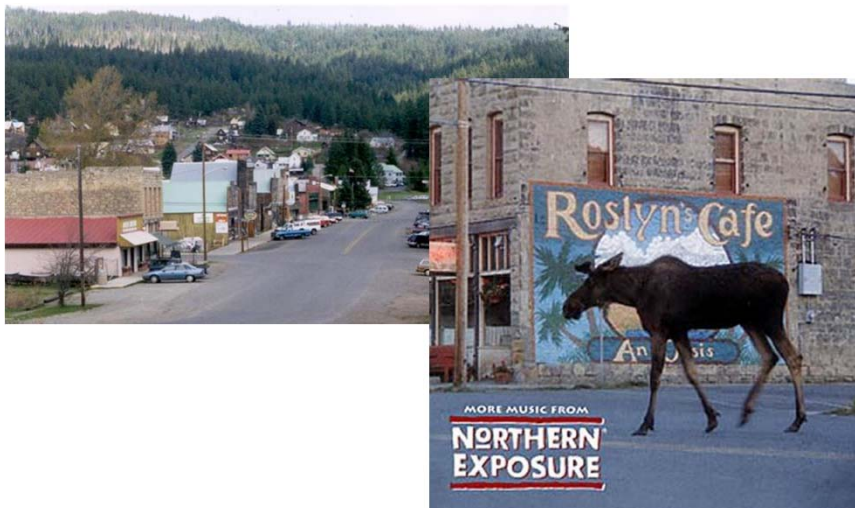
```
var script = CSharpScript.Create<int>("int x = 1;")  
    .ContinueWith("int y = 2;")  
    .ContinueWith("x + y");
```

```
Assert.AreEqual(  
    3,  
    (int)(await script.RunAsync()).ReturnValue);
```

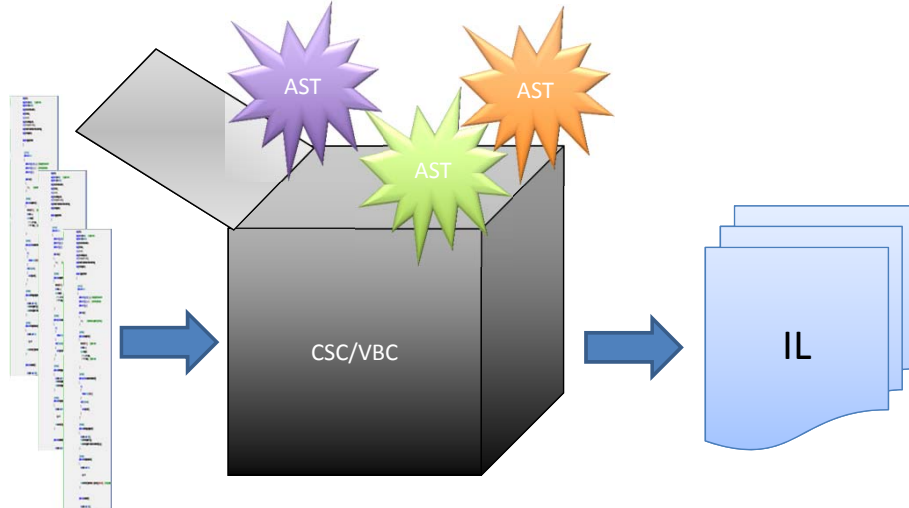
```
C# Interactive  
Microsoft (R) Roslyn C# Compiler version 1.2.0.60425  
Loading context from 'CSharpInteractive.rsp'.  
Type "#help" for more information.  
> var x = 1;  
> var y = 2;  
> x + y  
3  
>
```



What is Roslyn?



Opening up the compiler



Out of the Box Refactorings

Code Fixes (30)

- Add Await
- Add Async
- Change Return Type
- Change To Yield
- Convert To Async
- Convert To Iterator
- Add Missing Reference
- Add Using
- Fully Qualify
- Generate Method

Refactorings (13)

- Add Constructor Parameters from members
- Change Signature
- Encapsulate Field
- Extract Interface
- Extract Method
- Generate Constructor from members
- Generate default constructors
- Inline temporary
- Introduce Variable
- Invert If
- Simplify Lambda

Microsoft.CodeAnalysis.Internal.Log



Third Party Code Fixes

- [Roslyn Analyzers](#)
- [Code Cracker](#)
- [DotNetAnalyzers](#)
 - [StyleCop](#)
 - [AsyncUsageAnalyzers](#)
 - [TSqlAnalyzer](#)
- [DotNetDoodle](#)
- [RoslynDiagnostics](#)
- [AzureAnalytics](#)
- [Wintellect.Analyzers](#)
- [Xunit.Analyzers](#)



Building your own Diagnostic with Code Fix

```
public class TestControllerMisnamed : Controller
{
    public TestControllerMisnamed() {}

    public ActionResult Index()
    {
        return View();
    }
}
```



Building your own Fix

ExpressionSyntax
SyntaxToken
SyntaxTrivia

```
var value = 1 + 2;

var equalsValue =
    SyntaxFactory.EqualsValueClause(
        SyntaxFactory.BinaryExpression(
            SyntaxKind.AddExpression,
            SyntaxFactory.LiteralExpression(
                SyntaxKind.NumericLiteralExpression,
                SyntaxFactory.Literal(1)),
            SyntaxFactory.LiteralExpression(
                SyntaxKind.NumericLiteralExpression, SyntaxFactory.Literal(2))
        ));

var typeSyntax =
    SyntaxFactory.VariableDeclaration(
        SyntaxFactory.ParseTypeName("var"),
        new SeparatedSyntaxList<VariableDeclaratorSyntax>()
            .Add(SyntaxFactory.VariableDeclarator("value"))
            .WithInitializer(equalsValue))
        .NormalizeWhitespace(" ")
        .WithAdditionalAnnotations(Formatter.Annotation);

var declaration = SyntaxFactory.LocalDeclarationStatement(typeSyntax);
```



Custom Diagnostic - Analyzer

```
private void Analyzer(SymbolAnalysisContext context)
{
    var symbol = (INamedTypeSymbol)context.Symbol;
    if (symbol.BaseType == null) return;

    if ((symbol.BaseType.Name == "Controller"
        || symbol.BaseType.Name == "ApiController")
        && !symbol.Name.EndsWith("Controller"))
    {
        var diagnostic = Diagnostic.Create(
            Rule, symbol.Locations[0], symbol.Name);
        context.ReportDiagnostic(diagnostic);
    }
}
```



Custom Diagnostic - CodeFixProvider


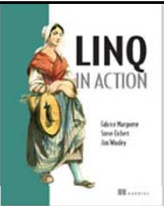
```
private async Task<Document> MakeEndInControllerAsync(  
    Document document,  
    TypeDeclarationSyntax typeDecl,  
    CancellationToken cancellationToken)  
{  
    var identifierToken = typeDecl.Identifier;  
    var originalName = identifierToken.Text;  
  
    var nameWithoutController = Regex.Replace(  
        originalName, "controller", String.Empty, RegexOptions.IgnoreCase);  
    var newName = nameWithoutController + "Controller";  
  
    var newIdentifier = SyntaxFactory.Identifier(newName)  
        .WithAdditionalAnnotations(Formatter.Annotation);  
  
    var newDeclaration = typeDecl.ReplaceToken(identifierToken, newIdentifier);  
  
    var root = await document.GetSyntaxRootAsync();  
    var newRoot = root.ReplaceNode(typeDecl, newDeclaration);  
    var newDocument = document.WithSyntaxRoot(newRoot);  
    return newDocument;  
}
```



Roslyn Resources

- Github: <https://github.com/dotnet/roslyn>
- Dotnet: <https://github.com/dotnet/roslyn>
- Reference Source: <http://source.roslyn.io/>
- Code Cracker: <http://code-cracker.github.io/>
- This presentation:
[HTTPS://GITHUB.COM/JWOOLEY/ROSLYNANDYOU](https://github.com/jwooley/RoslynAndYou)





C# 7 (+) ROSLYN AND YOU

<https://github.com/jwooley/RoslynAndYou>
<https://jwooley.github.io/>

Jim Wooley
www.ThingLinq.com
Twitter: @JimWooley

