

What's New in C# 6.0

By Daniel D'Agostino
29th January 2015

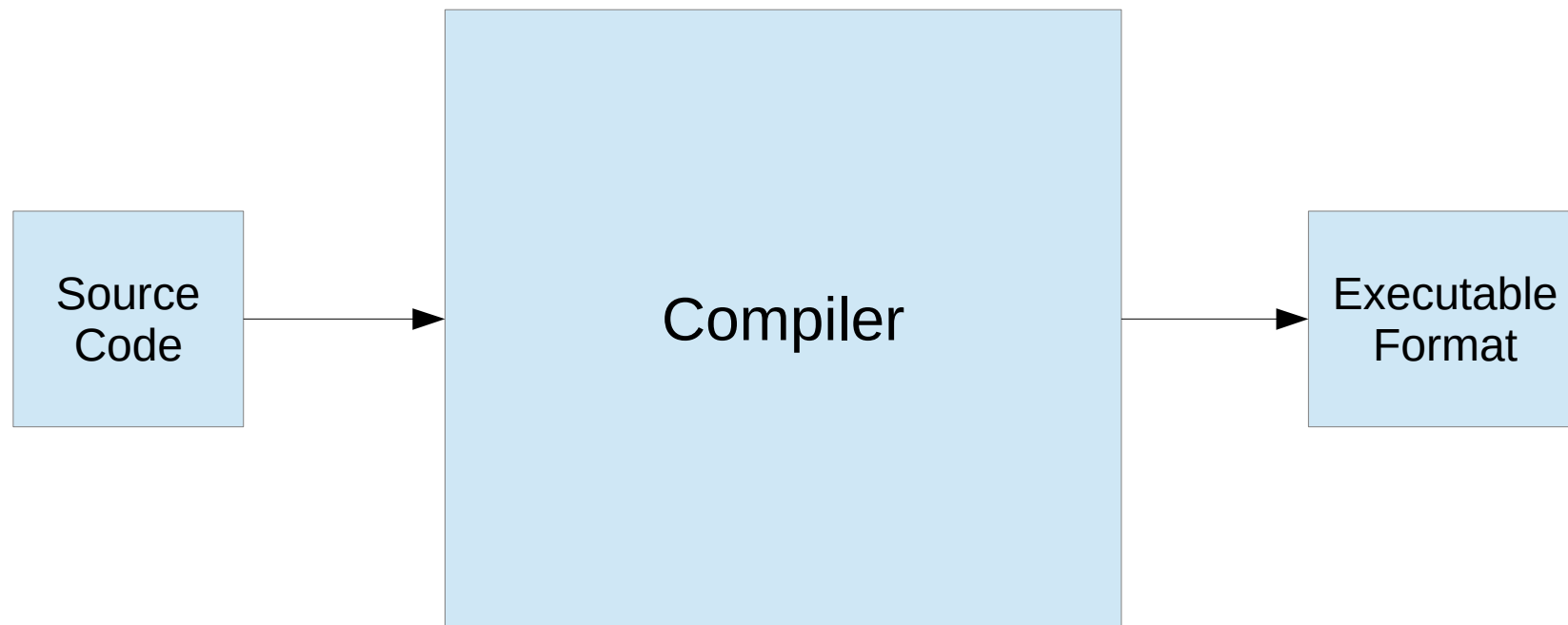


Visual Studio 2015 Preview

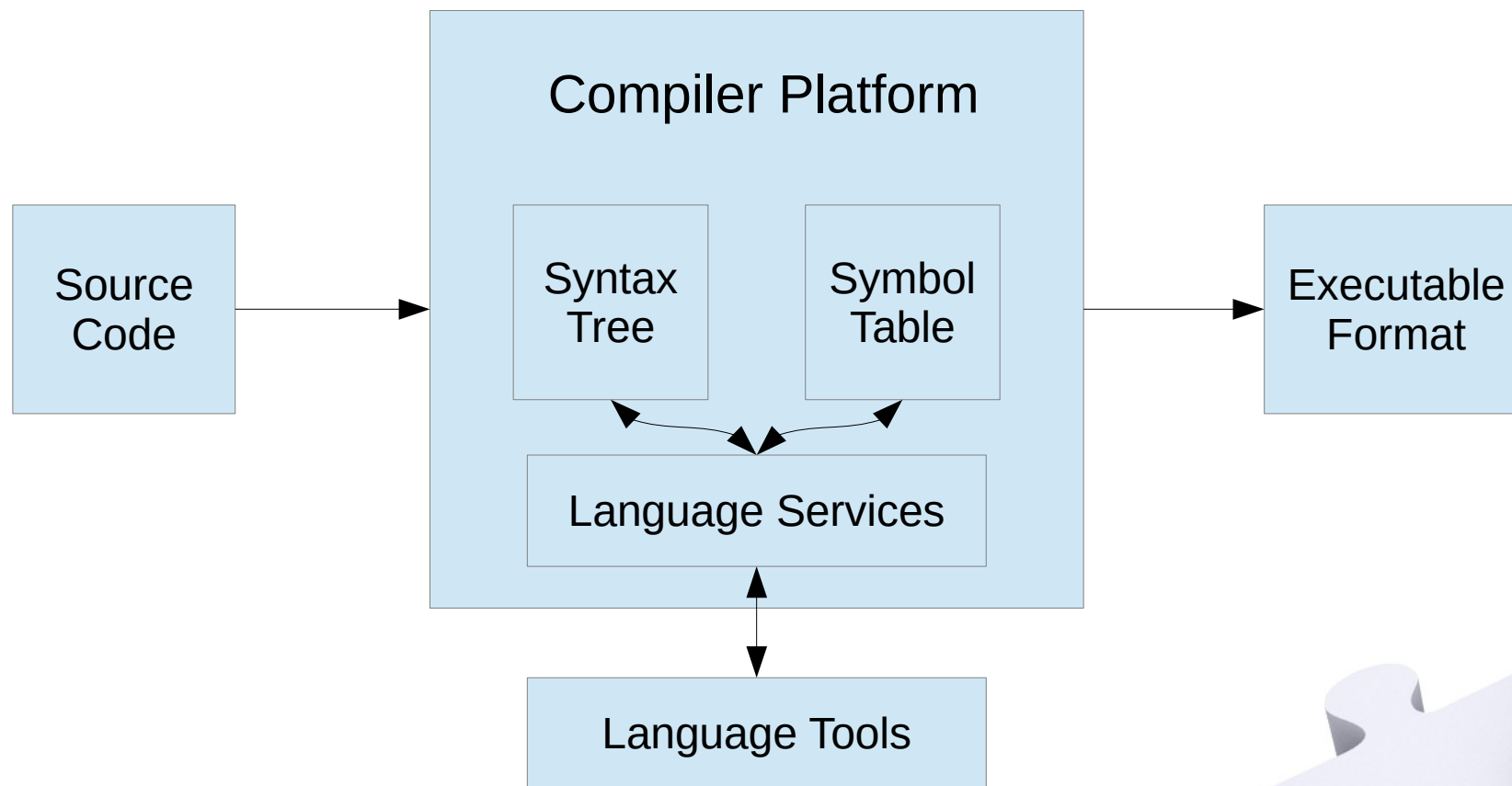
- Next version of .NET
- ASP .NET vNext
- C++ language and debugging improvements
- .NET Compiler Platform (“Roslyn”)
 - IDE enhancements
 - New VB .NET features
 - New C# features



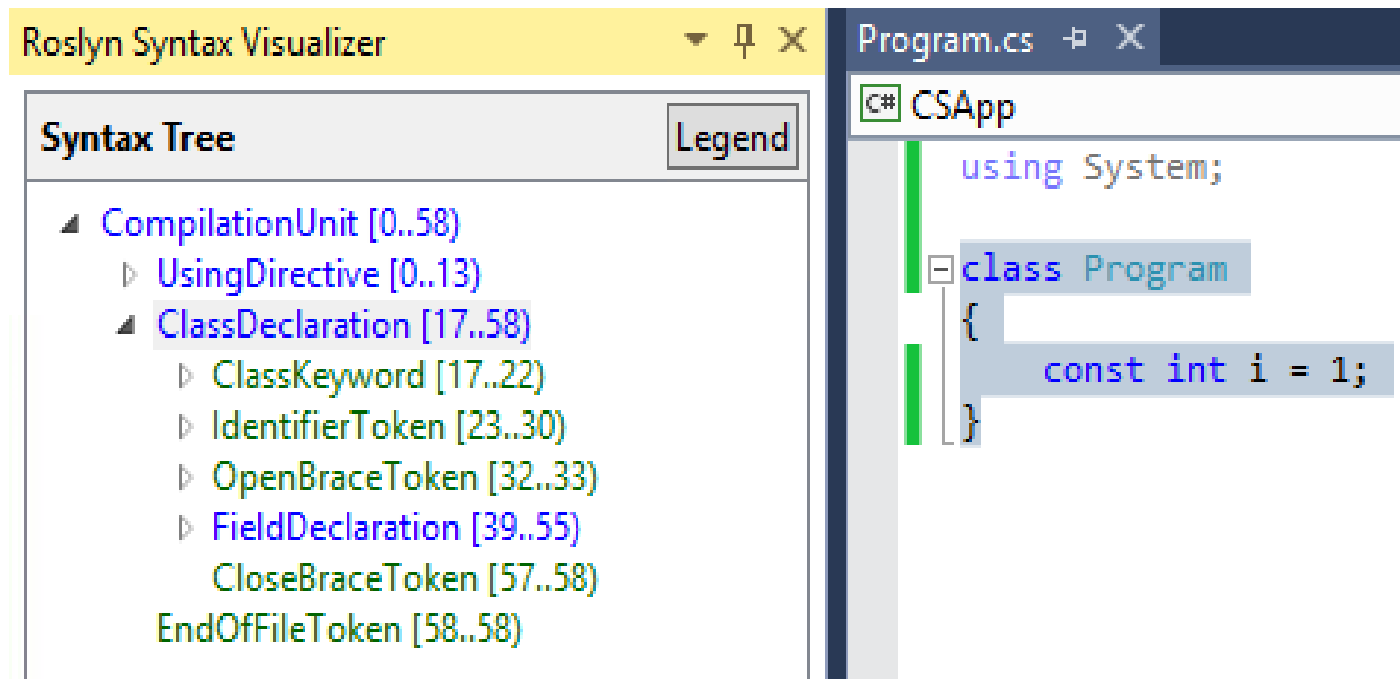
Traditional Compilers



Roslyn – Compiler As A Service



Roslyn Syntax Visualizer

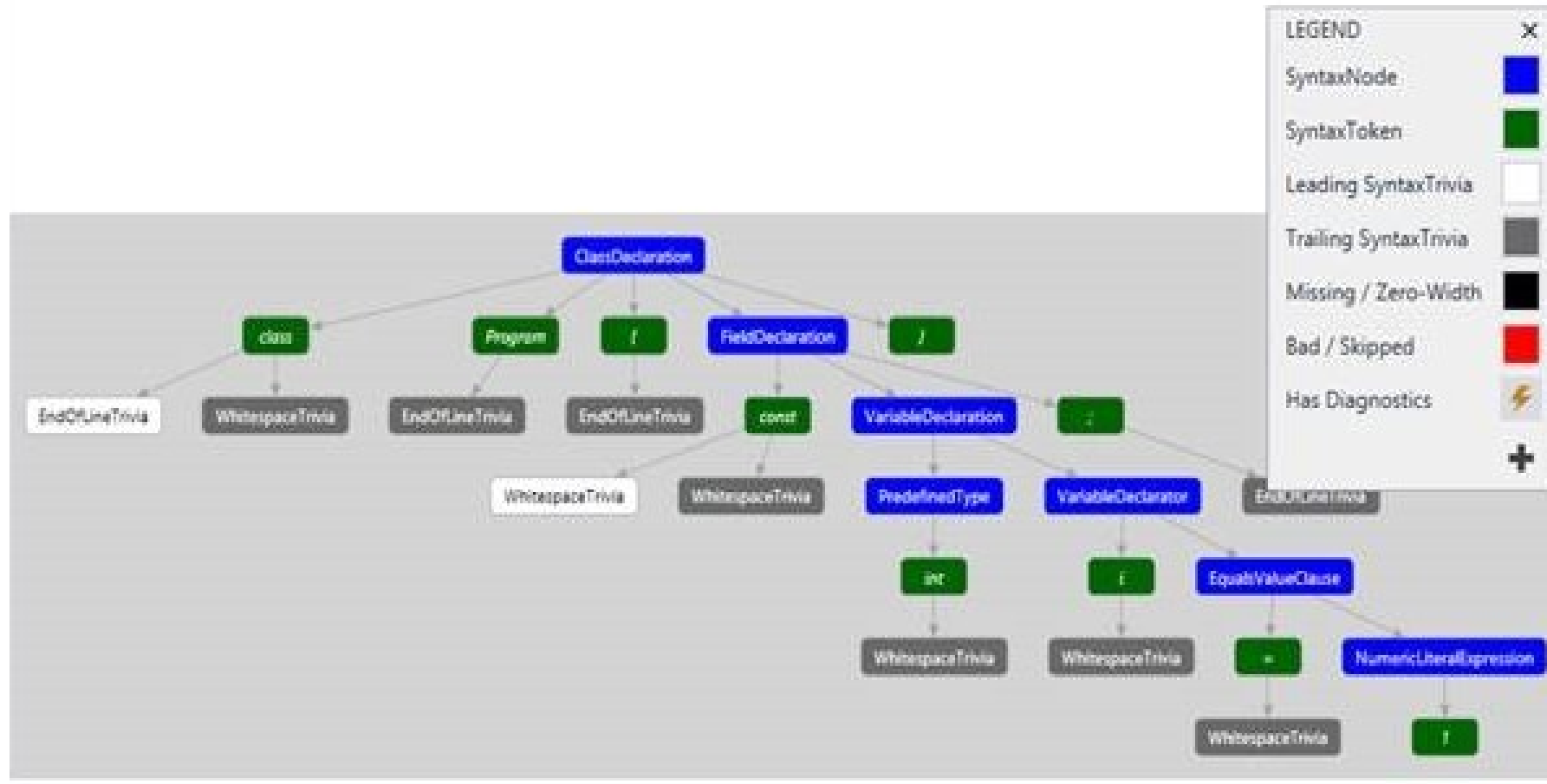


The screenshot displays the Roslyn Syntax Visualizer interface. On the left, the 'Syntax Tree' pane shows a tree structure for the code. The root node is 'CompilationUnit [0..58]', which contains a 'UsingDirective [0..13]' and a 'ClassDeclaration [17..58]'. The 'ClassDeclaration' node is expanded to show its children: 'ClassKeyword [17..22]', 'IdentifierToken [23..30]', 'OpenBraceToken [32..33]', 'FieldDeclaration [39..55]', 'CloseBraceToken [57..58]', and 'EndOfFileToken [58..58]'. On the right, the 'Program.cs' editor shows the corresponding C# code:

```
using System;  
  
class Program  
{  
    const int i = 1;  
}
```

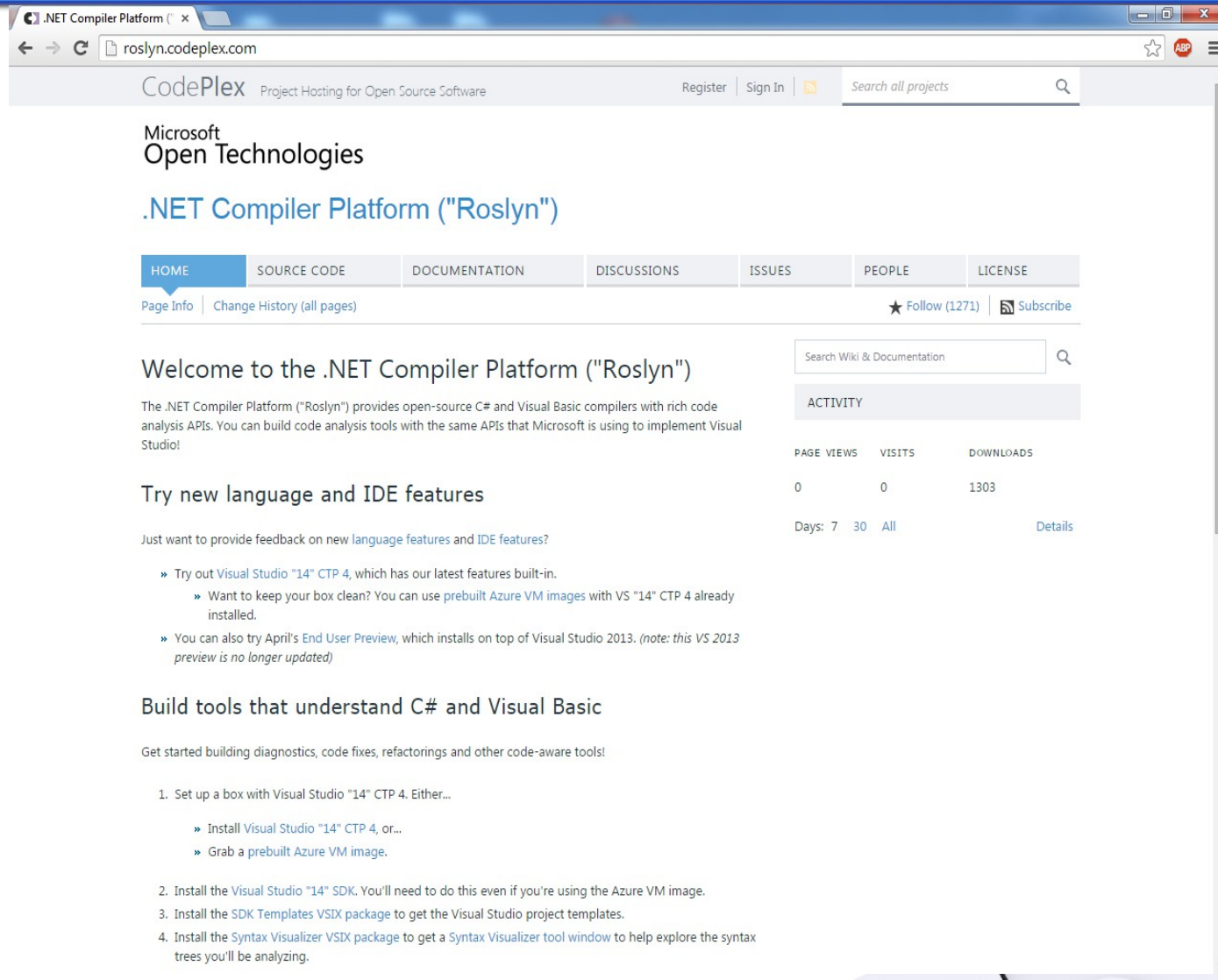
Source: <http://blogs.msdn.com/b/csharpfaq/archive/2014/04/17/visualizing-roslyn-syntax-trees.aspx>

Roslyn Syntax Visualizer



Source: <http://blogs.msdn.com/b/csharpfaq/archive/2014/04/17/visualizing-roslyn-syntax-trees.aspx>

Roslyn – The .NET Compiler Platform



The screenshot shows the CodePlex website for the .NET Compiler Platform project. The browser address bar shows 'roslyn.codeplex.com'. The page header includes the CodePlex logo and navigation links for Register, Sign In, and a search bar. The main content area features a navigation menu with tabs for HOME, SOURCE CODE, DOCUMENTATION, DISCUSSIONS, ISSUES, PEOPLE, and LICENSE. Below the navigation menu, there are links for Page Info and Change History, and a 'Follow (1271)' button with a 'Subscribe' icon. The main heading is 'Welcome to the .NET Compiler Platform ("Roslyn")', followed by a paragraph describing the platform's capabilities. A section titled 'Try new language and IDE features' includes a link to provide feedback and a list of options for trying out Visual Studio 14 CTP 4. A section titled 'Build tools that understand C# and Visual Basic' includes a link to get started building diagnostics and a numbered list of steps for installation.

CodePlex Project Hosting for Open Source Software

Register | Sign In | Search all projects

Microsoft
Open Technologies

.NET Compiler Platform ("Roslyn")

HOME | SOURCE CODE | DOCUMENTATION | DISCUSSIONS | ISSUES | PEOPLE | LICENSE

Page Info | Change History (all pages) | Follow (1271) | Subscribe

Search Wiki & Documentation

ACTIVITY

PAGE VIEWS	VISITS	DOWNLOADS
0	0	1303

Days: 7 30 All Details

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

The .NET Compiler Platform ("Roslyn") provides open-source C# and Visual Basic compilers with rich code analysis APIs. You can build code analysis tools with the same APIs that Microsoft is using to implement Visual Studio!

Try new language and IDE features

Just want to provide feedback on new language features and IDE features?

- » Try out [Visual Studio "14" CTP 4](#), which has our latest features built-in.
 - » Want to keep your box clean? You can use [prebuilt Azure VM images](#) with VS "14" CTP 4 already installed.
- » You can also try [April's End User Preview](#), which installs on top of Visual Studio 2013. (*note: this VS 2013 preview is no longer updated*)

Build tools that understand C# and Visual Basic

Get started building diagnostics, code fixes, refactorings and other code-aware tools!

1. Set up a box with Visual Studio "14" CTP 4. Either...
 - » Install Visual Studio "14" CTP 4, or...
 - » Grab a [prebuilt Azure VM image](#).
2. Install the [Visual Studio "14" SDK](#). You'll need to do this even if you're using the Azure VM image.
3. Install the [SDK Templates VSIX package](#) to get the Visual Studio project templates.
4. Install the [Syntax Visualizer VSIX package](#) to get a [Syntax Visualizer tool window](#) to help explore the syntax trees you'll be analyzing.



Roslyn – The .NET Compiler Platform

- Open Source
- Compiler as a Service
 - API for language tools such as ReSharper
 - New IDE features
- New compilers for C# and VB .NET
 - New language features



What's New in C# 6.0?

- nameof() expressions
- using static
- await in catch/finally blocks
- Auto-property initializers
- Getter-only auto-properties
- Ctor assignment to getter-only autoprops



What's New in C# 6.0?

- Safe Navigation Operator (?.)
- Exception filters
- Parameterless struct constructors
- Expression-bodied members
- Index initializers
- String interpolation



nameof() expressions

- Demo: ArgumentNullException
- Demo: INotifyPropertyChanged



using static

- Demo: Console
- Demo: Math



await in catch/finally blocks

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
    using (var logger = new Logger())
    {
        try
        {
            await logger.LogAsync("Executing operation...");
        }
        catch (Exception ex)
        {
            await logger.LogAsync(ex.ToString());
        }
        finally
        {
            await logger.FlushAsync();
        }
    }
}
```



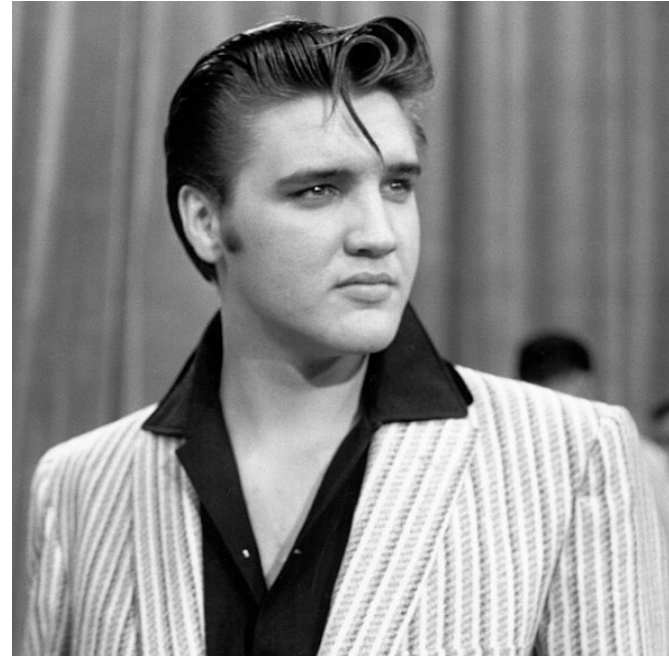
Autoproperty enhancements

- Demo: Autoproperty enhancements
- Auto-property initializers
- Getter-only auto-properties
- Ctor assignment to getter-only autoprops



Safe Navigation Operator (?.)

- Known as
 - Safe Navigation Operator
 - Null-Conditional Operator
 - Null Propagation Operator
 - Conditional Access Operator
 - Elvis Operator



Safe Navigation Operator (?.)

- Demo: Safe Navigation Operator (Properties)
 - Session Id
 - Null-coalesce
 - Indexers
- Demo: Safe Navigation Operator (Events)



Exception filters

```
try
{
    throw new InvalidOperationException("Invalid operation",
        new Exception("Something just exploded"));
}
catch (Exception ex) if (ex.InnerException != null)
{
    Console.WriteLine(ex.Message);
    Console.WriteLine(" -->{0}", ex.InnerException.Message);
}
catch (Exception ex)
{
    Console.WriteLine(ex);
}
```



Exception filters

“

It is also a common and accepted form of “abuse” to use exception filters for side effects; e.g. logging. They can inspect an exception “flying by” without intercepting its course. In those cases, the filter will often be a call to a false-returning helper function which executes the side effects:

```
private static bool Log(Exception e) { /* log it */ ; return false; }  
...  
try { ... } catch (Exception e) if (Log(e)) {}
```

”

Source: official C# feature descriptions
<http://www.codeplex.com/Download?ProjectName=roslyn&DownloadId=894944>

Parameterless struct constructors

- Demo: Point
- Still must initialize all members of the struct
- Must be public



Expression-bodied members

- Single-statement methods/properties
- Demo: Person
- Demo: Singleton
- Demo: Inventory
- Getter-only properties
- Methods with return values
- void methods



Index initializers

- Demo: Morse Code dictionary



String interpolation

- Demo: String Interpolation
 - Simple interpolation
 - Format strings



Resources

- C# 6 feature descriptions on my blog:
 - <http://gigi.nullneuron.net/gigilabs/tag/c-6/>
- Old Roslyn Codeplex homepage:
 - <http://roslyn.codeplex.com/>
- New Roslyn GitHub homepage:
 - <https://github.com/dotnet/roslyn>
- Visual Studio blog:
 - <http://blogs.msdn.com/b/visualstudio/>

