What's New in C# 6.0

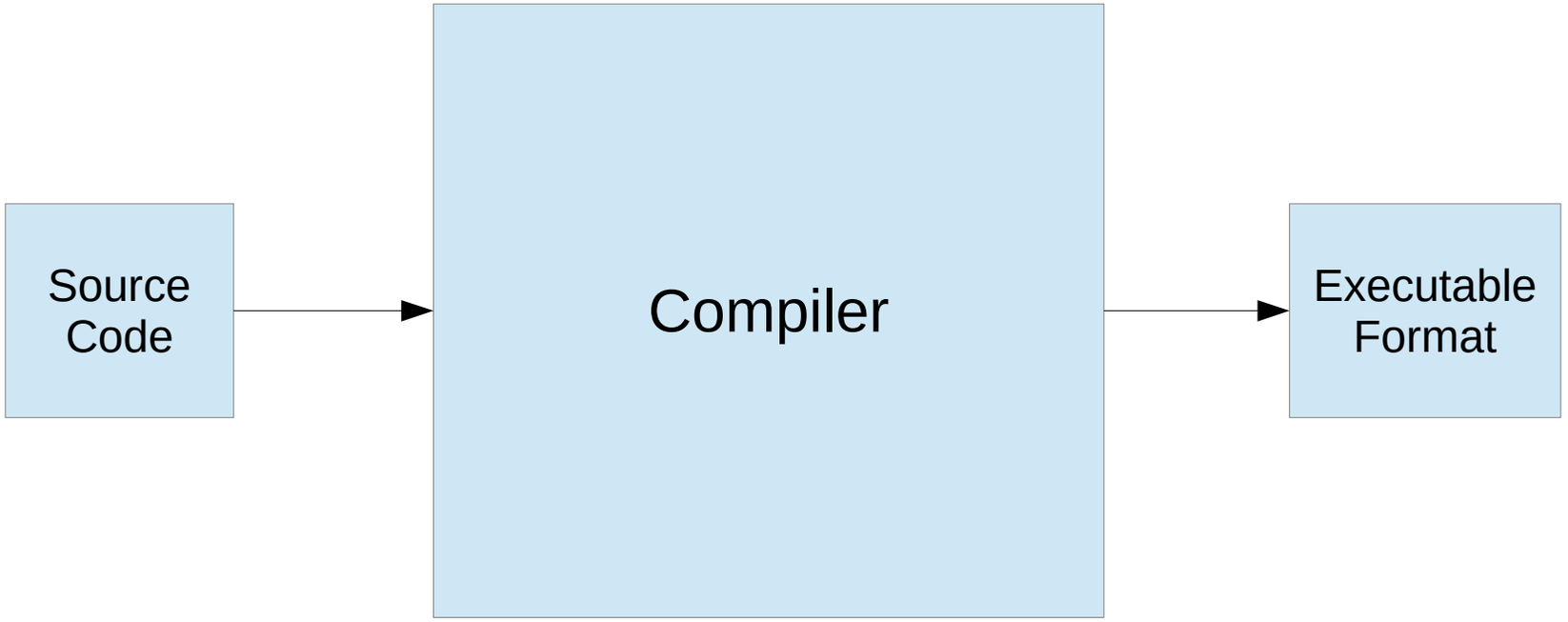
By Daniel D'Agostino

29th January 2015

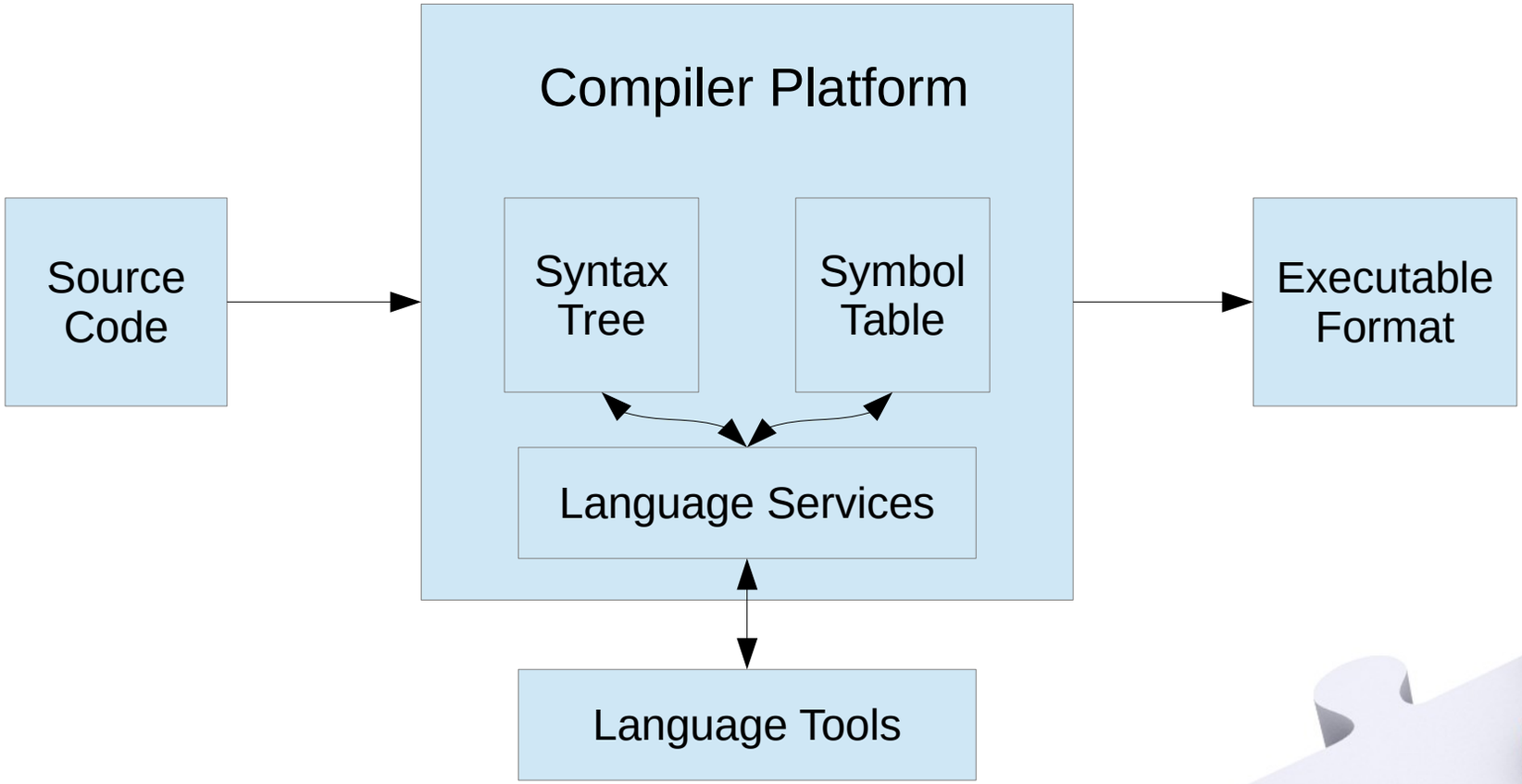
Visual Studio 2015 Preview

- 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

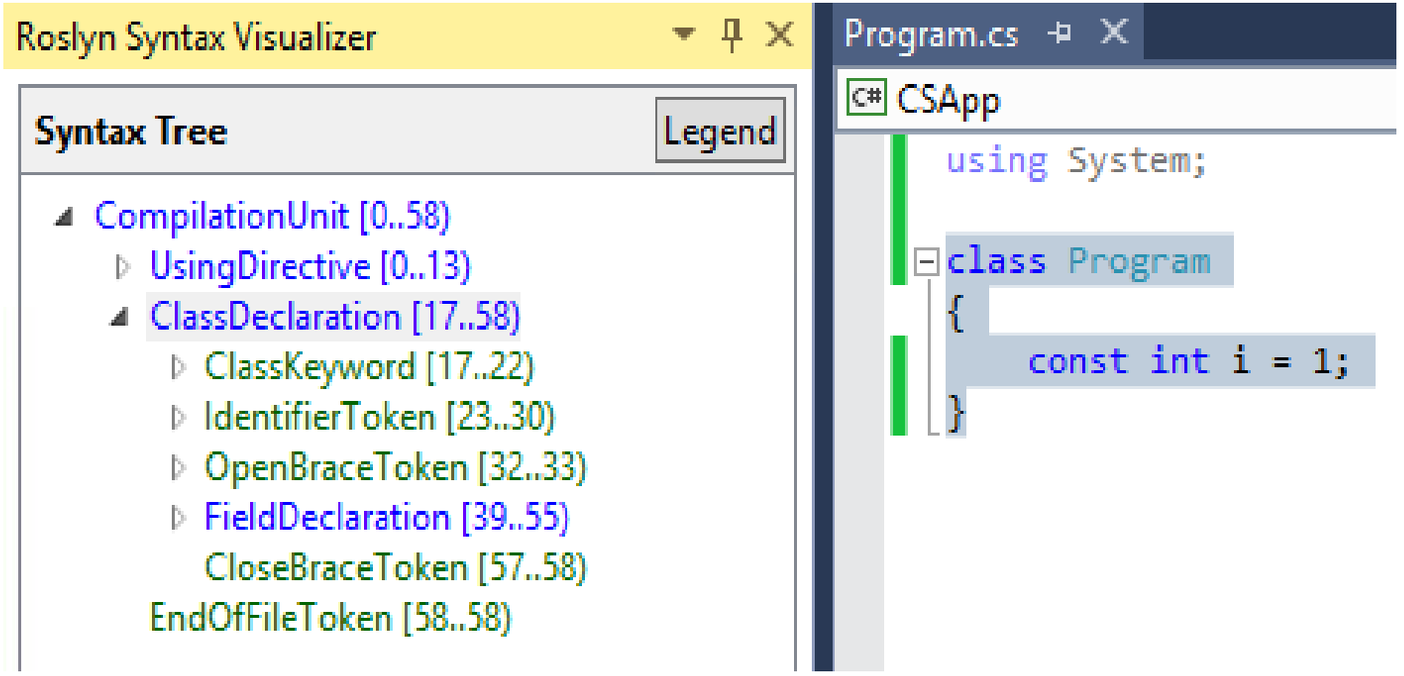
Traditional Compilers



Roslyn – Compiler As A Service

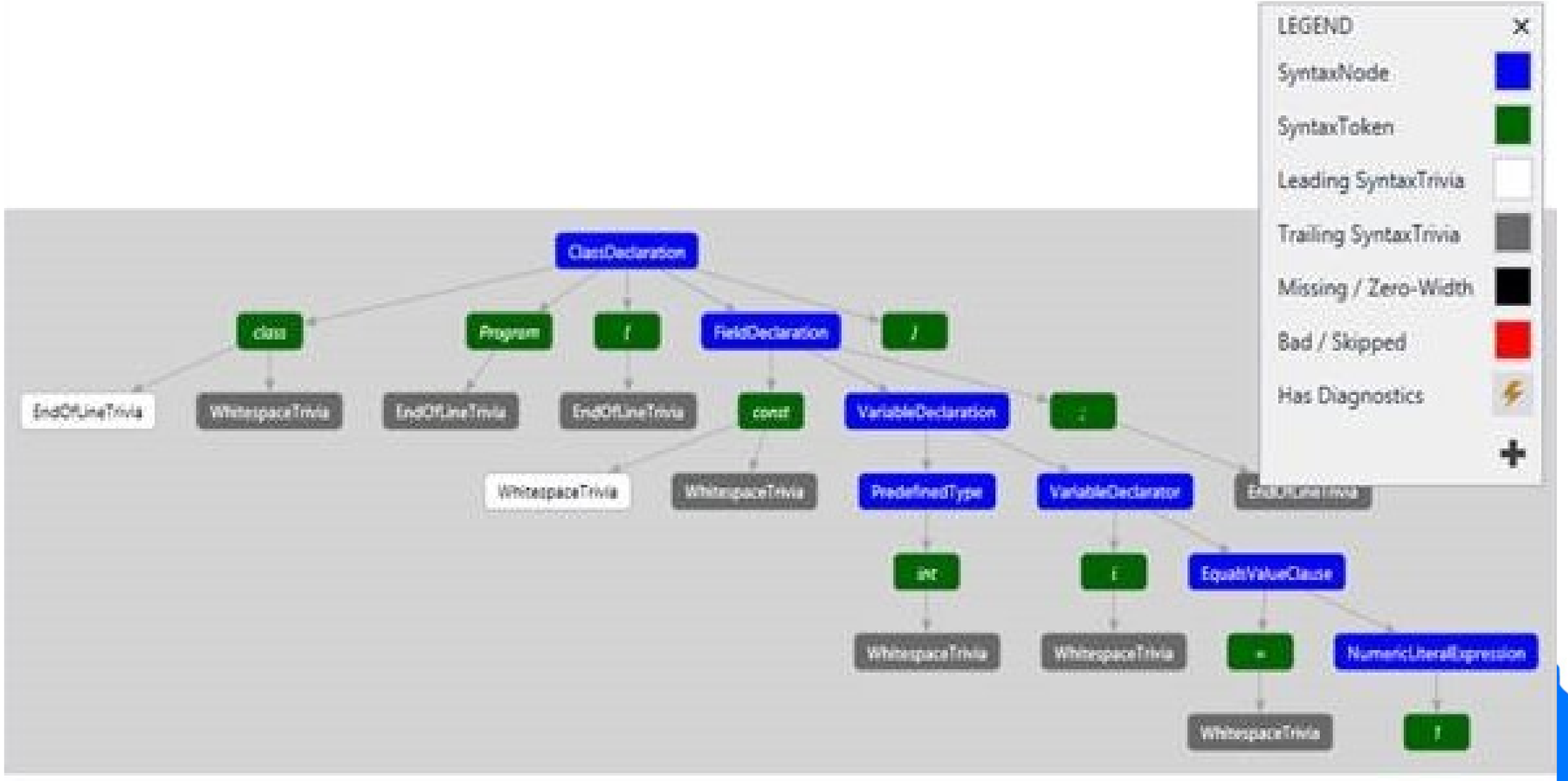


Roslyn Syntax Visualizer



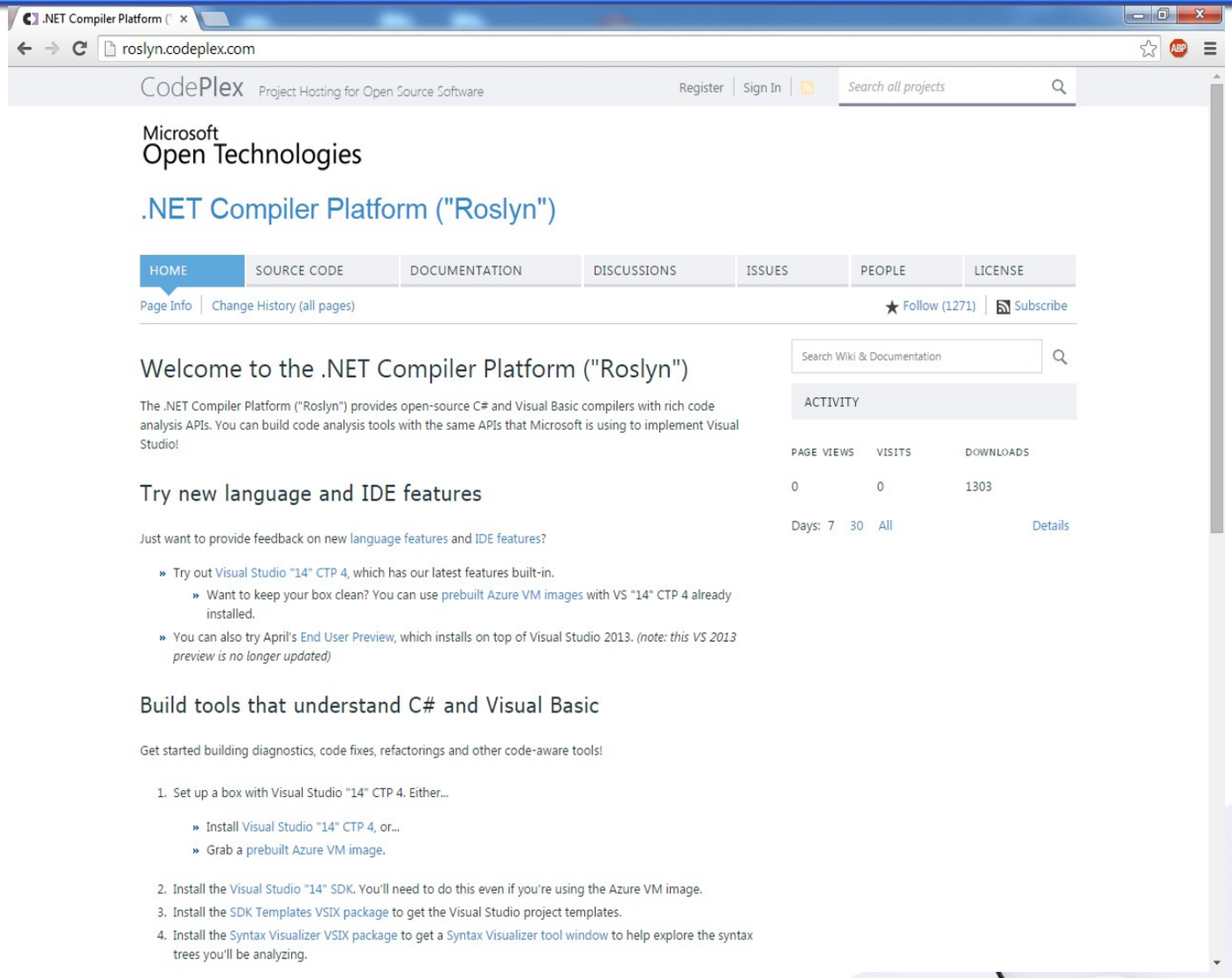
Source: http://blogs.msdn.com/b/csharpfaq/archive/2014/04/17/visualizing-roslyn-syntax-trees.aspx

Roslyn Syntax Visualizer



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

Roslyn – The .NET Compiler Platform



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

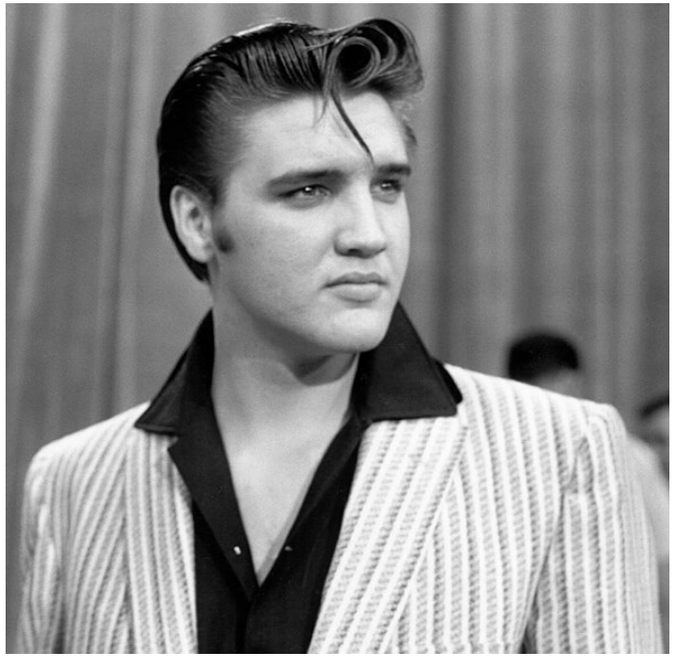
```
private async void Button_Click(object sender, RoutedEventArgs e)
{
    using (var logger = new Logger())
    {
        try
        {
            await logger.LogAsync("Executing operation...");
        }
        catch (Exception ex)
        {
            await logger.LogAsync(ex.ToString());
  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

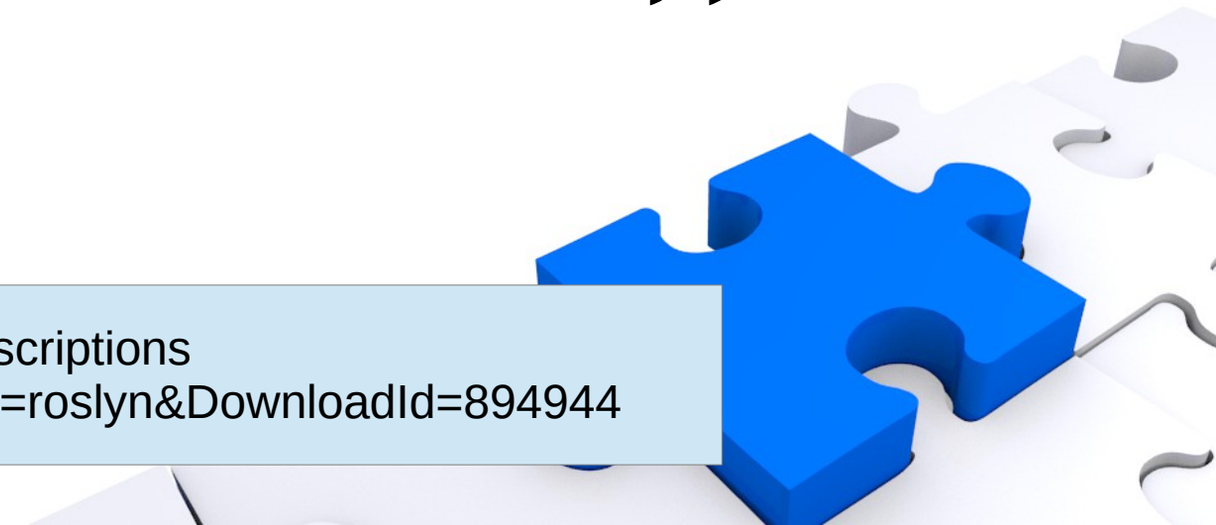
“

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/