



SELA|DEVELOPER|PRACTICE
JUNE 19-23, 2016

Avi Avni

Building Roslyn Analyzers and Code Fixes

Blog: http://blogs.microsoft.co.il/avi_avni

GitHub: <https://github.com/AviAvni/SDP2016>

Agenda

- ✦ What are **Analyzers** and **Code Fixes**?
 - ✦ What can we do with **Analyzers** and **Code Fixes**?
 - ✦ How to get **Analyzers** and **Code Fixes**?
 - ✦ What is **Roslyn**?
 - ✦ The **Roslyn** Architecture
 - ✦ **IOperation** – New API
-

What are Analyzers and Code Fixes?

- ✦ **Analyzers** are a way to write C# code to analyze C# code
 - ✦ **Code Fixes** are a way to write C# code to change C# code
 - ✦ Better way for **FxCop**
 - ✦ Run live in **Visual Studio**
-

What can we do with Analyzers and Code Fixes?

- ✦ Identify errors before compile time
 - ✦ Enforce code style
 - ✦ Introduce best practices for SDK
 - ✦ Teach new language feature/concepts
-

How to get Analyzers and Code Fixes?

	NuGet	Visual Studio Extensions
File Type	*.dll	*.vsix
Run on vs	Specific Project	Every solution
Run on build Machine	Yes	No
Samples	Desktop.Analyzers ApiDesignGuidelines ErrorProne.Net	C# Essential Clr Heap Allocation Analyzer

Live Analyzers and Code Fixes

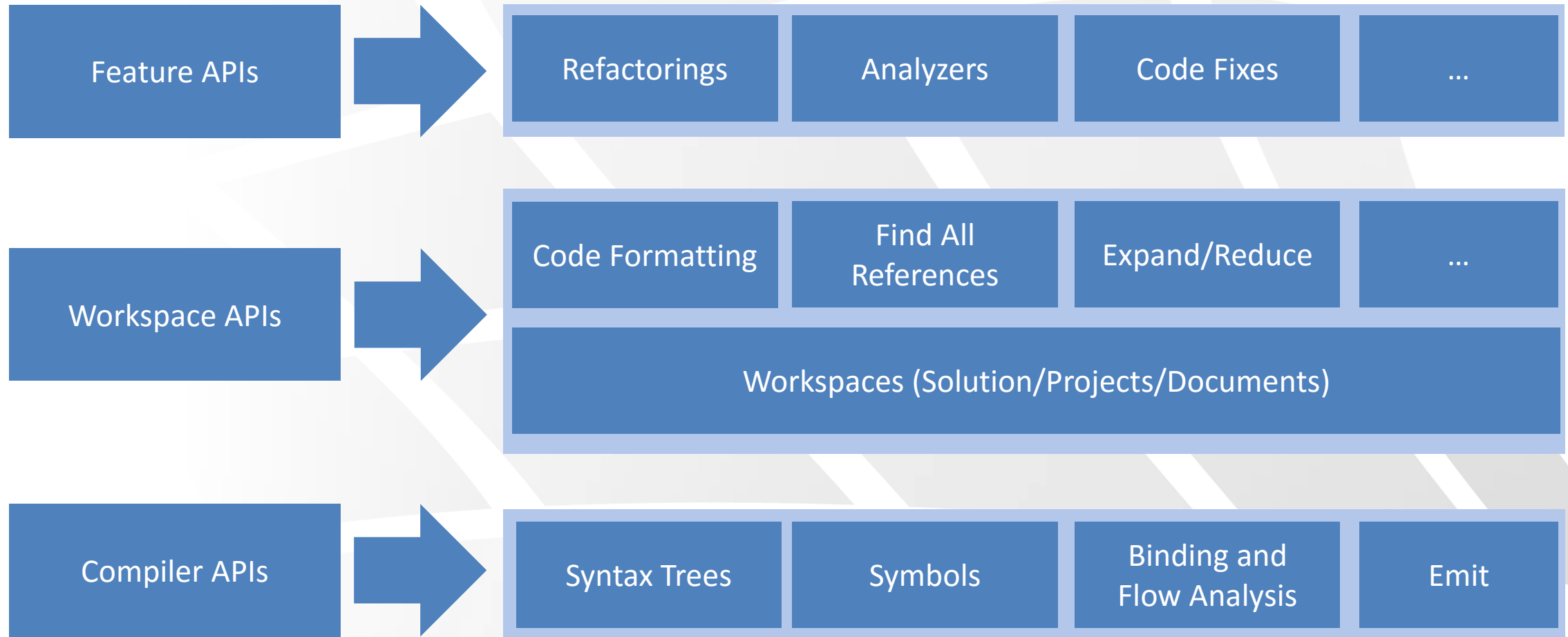
Demo



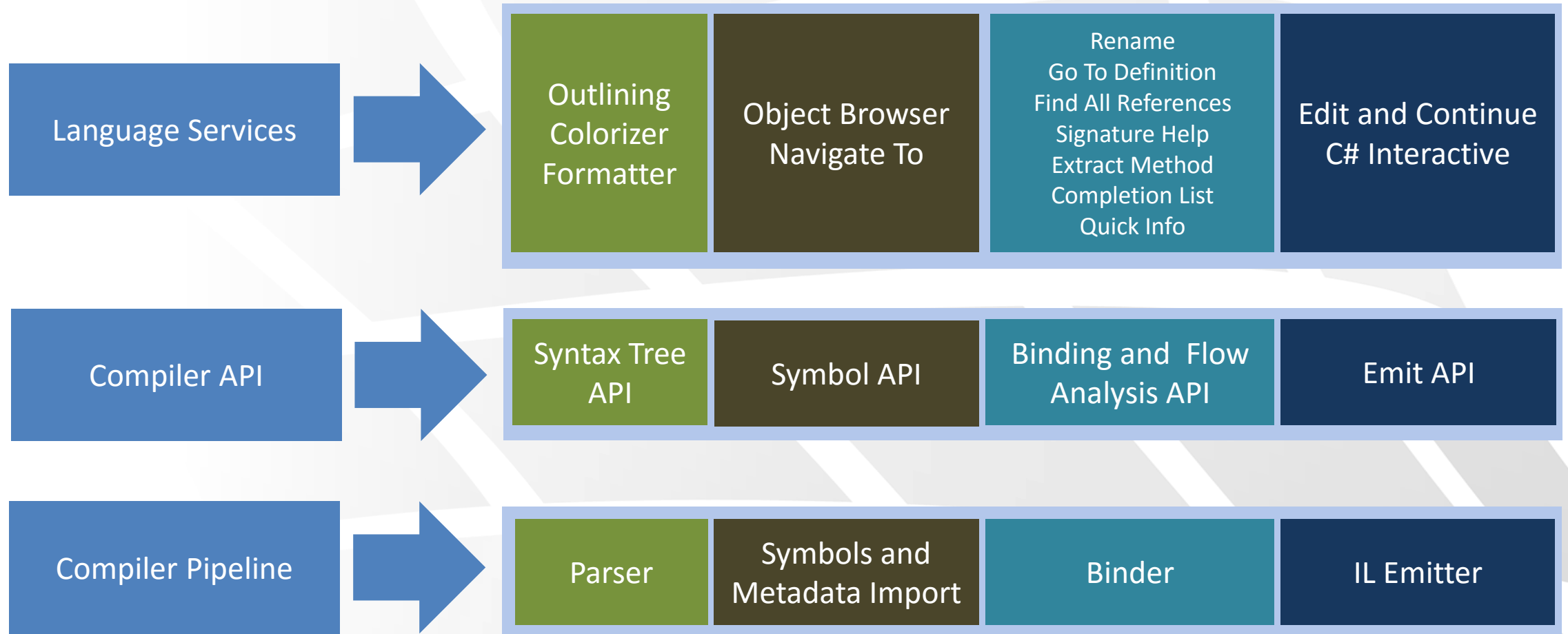
What is Roslyn?

- ✦ The .NET Compiler Platform
 - ✦ One code base for all C#/VB analysis
 - ✦ C#/VB Compiler written in C#/VB
 - ✦ Implementation of all language services used in IDEs
 - ✦ Open API to compiler knowledge
 - ✦ Open source <https://github.com/dotnet/roslyn>
-

The Roslyn Architecture



The Roslyn Architecture



The Roslyn Architecture - Parser

- ✦ Source is tokenized and parsed into AST
- ✦ Syntax tree contains every piece of information found in the source text
- ✦ ASTs are immutable and thread-safe

Outlining
Colorizer
Formatter

Syntax Tree
API

Parser

Syntax Visualizer

Syntax Visualizer

Syntax Tree

Legend

- IdentifierToken [167..174]
- OpenBraceToken [180..181]
- MethodDeclaration [191..392]
 - PublicKeyword [191..197]
 - PredefinedType [198..202]
 - IdentifierToken [203..206]
 - ParameterList [206..208]
 - Block [218..392]
 - OpenBraceToken [218..219]
 - TryStatement [233..381]
 - TryKeyword [233..236]
 - Block [250..290]
 - OpenBraceToken [250..251]
 - ExpressionStatement [269..275]
 - InvocationExpression [269..274]
 - IdentifierName [269..272]
 - ArgumentList [272..274]
 - SemicolonToken [274..275]
 - CloseBraceToken [289..290]
 - CatchClause [304..381]

Properties

Type	InvocationExpressionSyntax
Kind	InvocationExpression
ContainsDiagnostics	False
ContainsDirectives	False

Program.cs*

ConsoleApplication9

ConsoleApplication9.MyClass

Foo()

```
7 namespace ConsoleApplication9
8 {
9     0 references
10    class MyClass
11    {
12        1 reference
13        public void Foo()
14        {
15            Foo();
16        }
17        catch (Exception ex)
18        {
19            throw ex;
20        }
21    }
22    1 reference
```

The Roslyn Architecture - Symbols

- ✦ Declarations from source and imported metadata are analyzed to form named symbols

Declared At	Type	Name
ConsoleApplication.dll	Class	Program
Program	Method	Main
Mscorelib.dll	Class	Console
Console	Method	WriteLine

Object Browser
Navigate To

Symbol API

Symbols and
Metadata Import

The Roslyn Architecture - Binder

- ✦ Identifiers in the code are matched to symbols
- ✦ Data Flow Analysis
- ✦ Control Flow Analysis

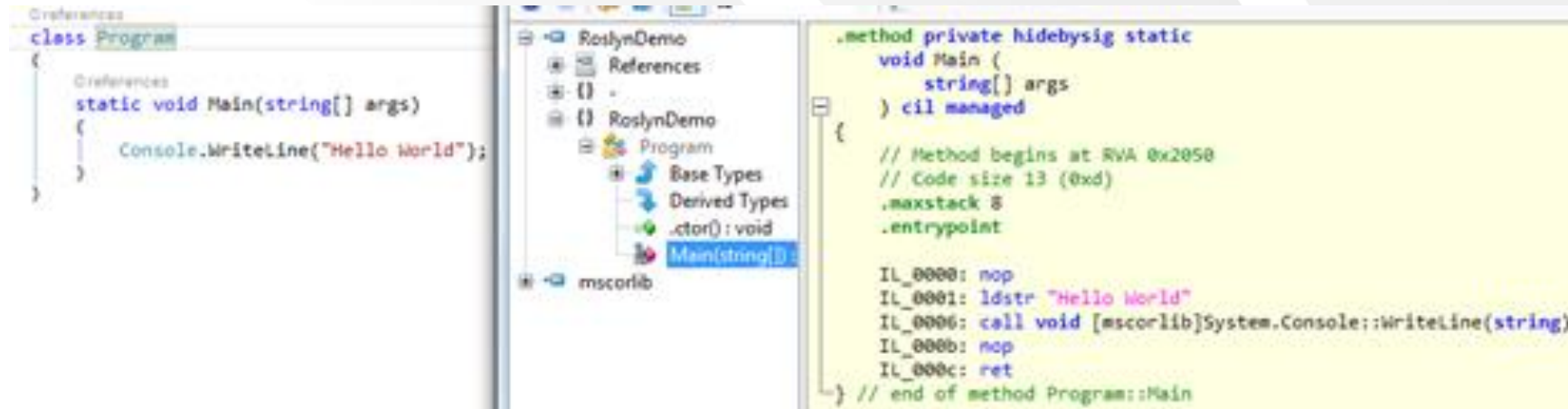
Rename
Go To Definition
Find All References
Signature Help
Extract Method
Completion List
Quick Info

Binding and Flow
Analysis API

Binder

The Roslyn Architecture – IL Emitter

- ✦ All the information built up by the compiler is emitted as an assembly
- ✦ C# Interactive and Scripting API





Edit and Continue
C# Interactive


Emit API

IL Emitter

Demo – Make Uppercase

18  1 reference
19 `class Program`

20 **Make uppercase** 

21 **Suppress DemoAnalyzer** 

22 `Foo(n`

23 `}`

24

25 `static st` 0 references

26 `{`

27 `try`

28 `{`

29 `r`

30 `}`

! DemoAnalyzer Type name 'Program' contains lowercase letters

...
//[Serializable]
`class Program`
`class PROGRAM`
{
 static void Main(string[] args)
 {
 Foo(new `Program`());
 Foo(new `PROGRAM`());
 }
...

Preview changes
Fix all occurrences in: [Document](#) | [Project](#) | [Solution](#)

Build Make Uppercase Analyzers and Code Fixes

Demo



Demo – Rethrow

```
25 static string Foo()  
26 {  
27     try  
28     {  
29         return "42";  
30     }  
31     catch (Exception ex)  
32     {
```

```
33         throw ex;  
34     }  
35 }  
36
```

Make rethrow

Suppress RethrowDemoAnalyzer

1 reference

```
37 static string F  
38 {  
39     try  
40     {  
41
```

⌵ ⛔ RethrowD
stack information

```
...  
{  
    throw ex;  
    throw;  
}  
...
```

- OpenBraceToken [210..210]
- TryStatement [233..381]
 - TryKeyword [233..236]
 - Block [250..290]
 - CatchClause [304..381]
 - CatchKeyword [304..309]
 - CatchDeclaration [310..324]
 - Block [338..381]
 - OpenBraceToken [338..339]
 - ThrowStatement [357..366]
 - ThrowKeyword [357..362]
 - IdentifierName [363..365]
 - IdentifierToken [363..365]
 - SemicolonToken [365..366]
 - CloseBraceToken [380..381]

Build Rethrow Analyzers and Code Fixes

Demo



Demo – NeedsAttribute

1 reference

```
class Program
```

```
{
```

0 references

```
static void Main(string[] args)
```

```
{
```

```
    Foo(new Program());
```

```
}
```

1 reference

```
static void Foo([Needs(typeof(SerializableAttribute))]object obj)
```

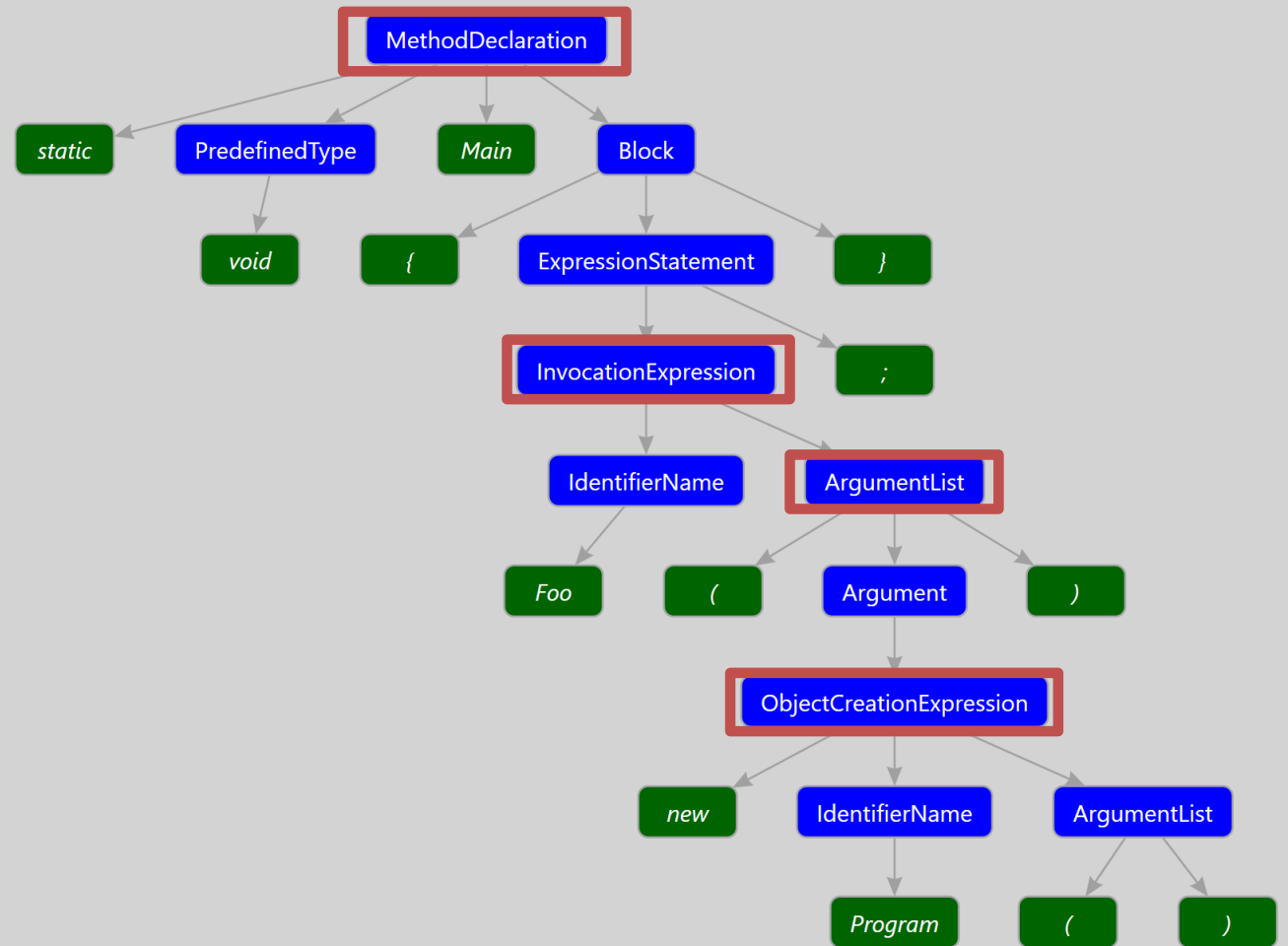
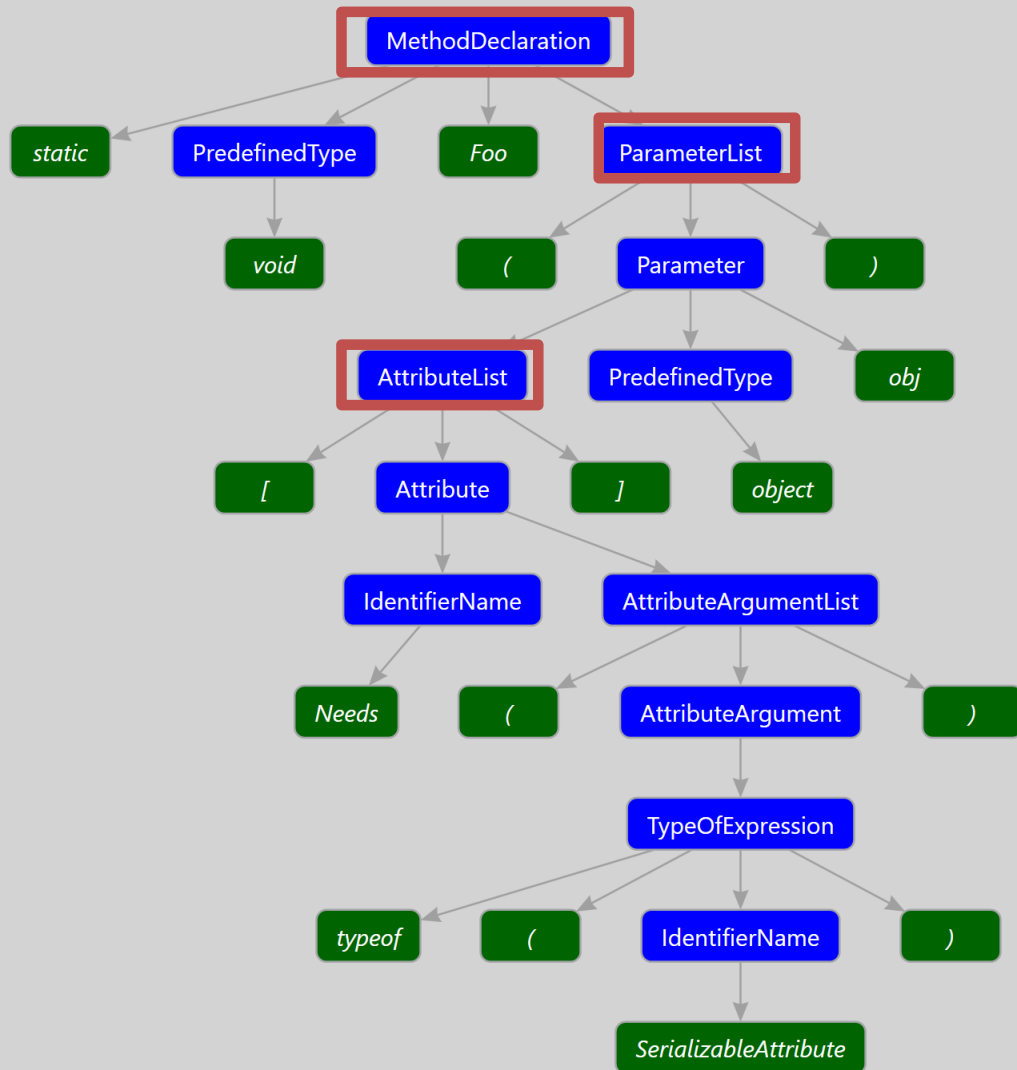
```
{
```

```
    if (!obj.GetType().IsDefined(typeof(SerializableAttribute), false))
```

```
        throw new ArgumentException("Argument must have Serializable attribute");
```

```
}
```

Demo – NeedsAttribute



Build NeedsAttribute Analyzer

Demo



IOperation

- ✦ In development
 - ✦ New API (VS15 Preview)
 - ✦ Exposes the bound tree
 - ✦ Cross language
 - ✦ Abstracts different language structures
-

Build NeedsAttribute Analyzer with IOperation API

Demo



Summary

- ✦ Analyzers can be help
 - ✦ Not every analyzer has to have code fixes
 - ✦ The Roslyn compiler phases
 - ✦ The Roslyn architecture
 - ✦ Roslyn is the basis for other tools
-

Resources

- ✦ <https://github.com/dotnet/roslyn/wiki/Roslyn%20Overview>
 - ✦ <https://msdn.microsoft.com/en-us/magazine/dn879356>
 - ✦ <https://msdn.microsoft.com/en-us/magazine/dn904670.aspx>
 - ✦ <https://github.com/dotnet/roslyn-analyzers>
 - ✦ <https://www.youtube.com/watch?v=zxAKyiQ1XiM>
 - ✦ <https://joshvarty.wordpress.com>
 - ✦ <https://channel9.msdn.com/Events/dotnetConf/2016/The-Power-of-Roslyn-Improving-Your-Productivity-with-Live-Code-Analyzers>
-

Questions

