



MODEL-BASED SOFTWARE DEVELOPMENT

PRACTICE 04 COMPILER AS A SERVICE

DR MEZEI GERGELY

BLACK BOX – WHITE BOX

- How does a code editor work? (e.g. IntelliJ)
 - We edit the code, we have
 - Syntax highlight
 - Syntax error visualization
 - Code completion
 - We run the code

The process of compilation is as a black box

BLACK BOX – WHITE BOX

- Compiler, as a black box
 - What happens if the code has errors? (syntax highlight)
 - How can we extend the compiler?
 - In many cases, it is enough, but...
 - The editor environment does not understand the inner mechanisms of the underlying compiler
 - AI-driven code completion? (e.g. Github Copilot)

BLACK BOX – WHITE BOX

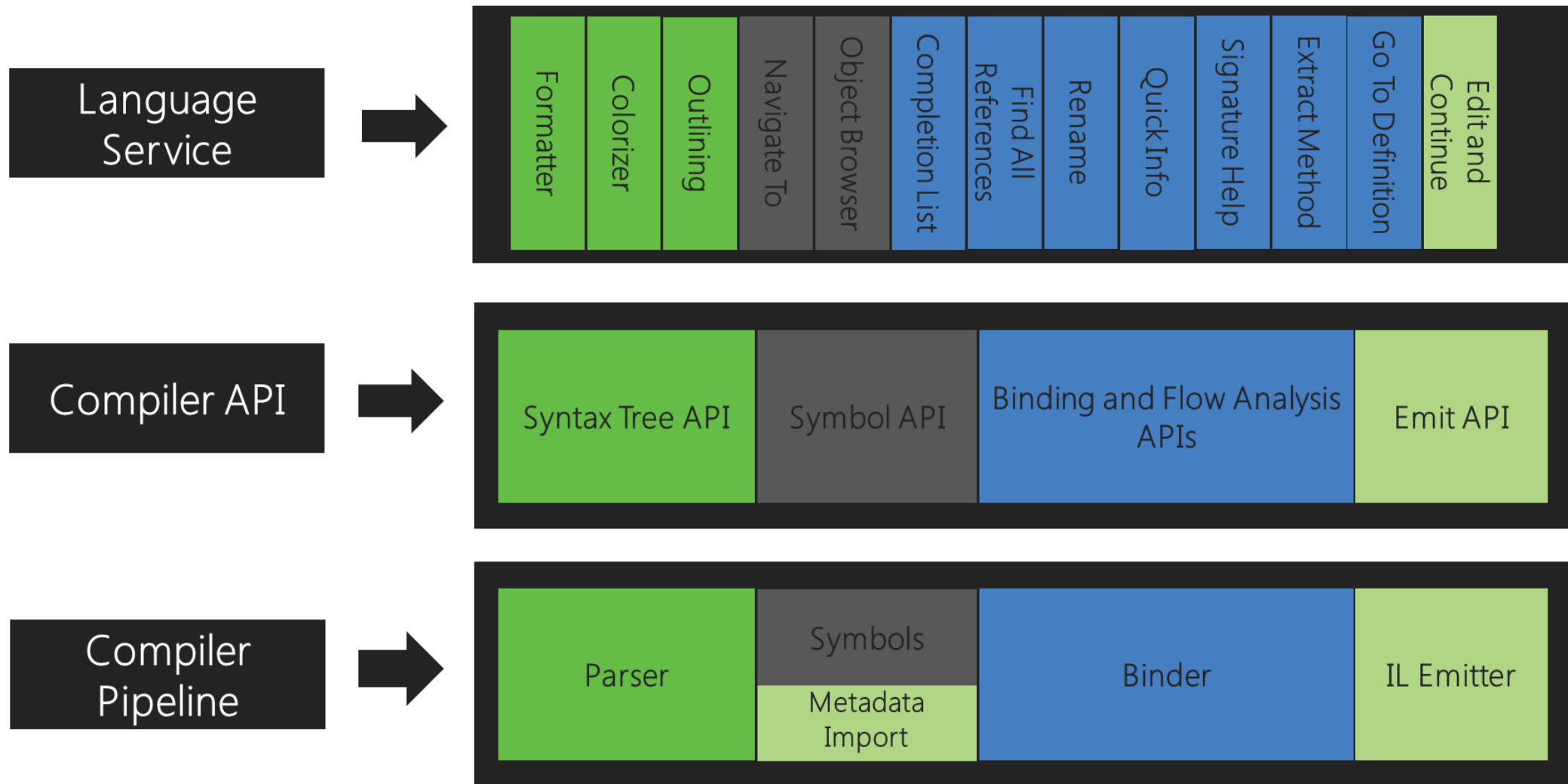
- Compiler – as a white box?
- Roslyn – Compiler as a Service
 - Code analyzer
 - Code fixer
 - Refactoring
 - Code generation
 - Built into Visual Studio

```
static void Main(string[] args)
{
    int i = 1;
    int j = 2;
    int k = i + j;
}
```

COMPILER AS A SERVICE

- CaaS
 - Custom (team-based) code conventions
 - E.g. Usage of camel case variables
 - Hints for using libraries/APIs
 - Help for general conventions, patterns

ROSLYN - COMPONENTS



DEMO: SYNTAX ANALYZER

Syntax Visualizer

Syntax Tree Legend

- UsingDirective [0..13]
- NamespaceDeclaration [17..189]
 - NamespaceKeyword [17..26]
 - IdentifierName [27..39]
 - OpenBraceToken [41..42]
- ClassDeclaration [48..186]
 - ClassKeyword [48..53]
 - IdentifierToken [54..61]
 - OpenBraceToken [67..68]
- MethodDeclaration [78..179]

Properties

Type MethodDeclarationSyntax
Kind MethodDeclaration

AttributeLists
Body { Console.Writel
ConstraintCla
ContainsAnnotz False
ContainsDiagn False
ContainsDirecti False
ContainsSkippe False

Program.cs

ConsoleApp12

ConsoleApp12.Program

Main(string[] args)

```
6 {  
7     0 references  
8     static void Main(string[] args)  
9     {  
10        Console.WriteLine("Hello World!");  
11    }  
12 }  
13
```

100 %

Syntax.dgml

Undo Show Related Layout Share 46.37% Legend Filters

```
graph TD
    MDN[MethodDeclaration Node] --> ST[Static Token]
    MDN --> PNTN[PredefinedType Node]
    MDN --> MT[Main Token]
    MDN --> PLN[ParameterList Node]
    MDN --> BN[Block Node]
    PLN --> PN[Parameter Node]
    PLN --> EOL1[EndOfLineTrivia]
    BN --> OBT[OpenBraceToken]
    BN --> ENSN[ExpressionStatement Node]
    BN --> EOL2[EndOfLineTrivia]
    ENSN --> INEN[InvocationExpression Node]
    ENSN --> ST2[SemicolonToken]
    INEN --> SMANEN[SimpleMemberAccessExpression Node]
    INEN --> AN[ArgumentList Node]
    SMANEN --> IDN1[IdentifierName Node]
    SMANEN --> IDN2[IdentifierName Node]
    AN --> ARGN[Argument Node]
    AN --> EOL3[EndOfLineTrivia]
    ARGN --> SLT[StringLiteralToken]
    ARGN --> EOL4[EndOfLineTrivia]
    SLT --> ST3[String Token]
    SLT --> EOL5[EndOfLineTrivia]
    ST3 --> ST4[String Token]
    ST3 --> EOL6[EndOfLineTrivia]
```

DEMO: DATA FLOW, CONTROL FLOW

- Artifacts of the semantic analysis can be used
 - Data flow
 - What symbols are defined?
 - Which symbols are available? What are their scopes? Name resolution
 - Is the given symbol (variable) used at all?
 - Control flow
 - Possible return values (not type!) of the methods

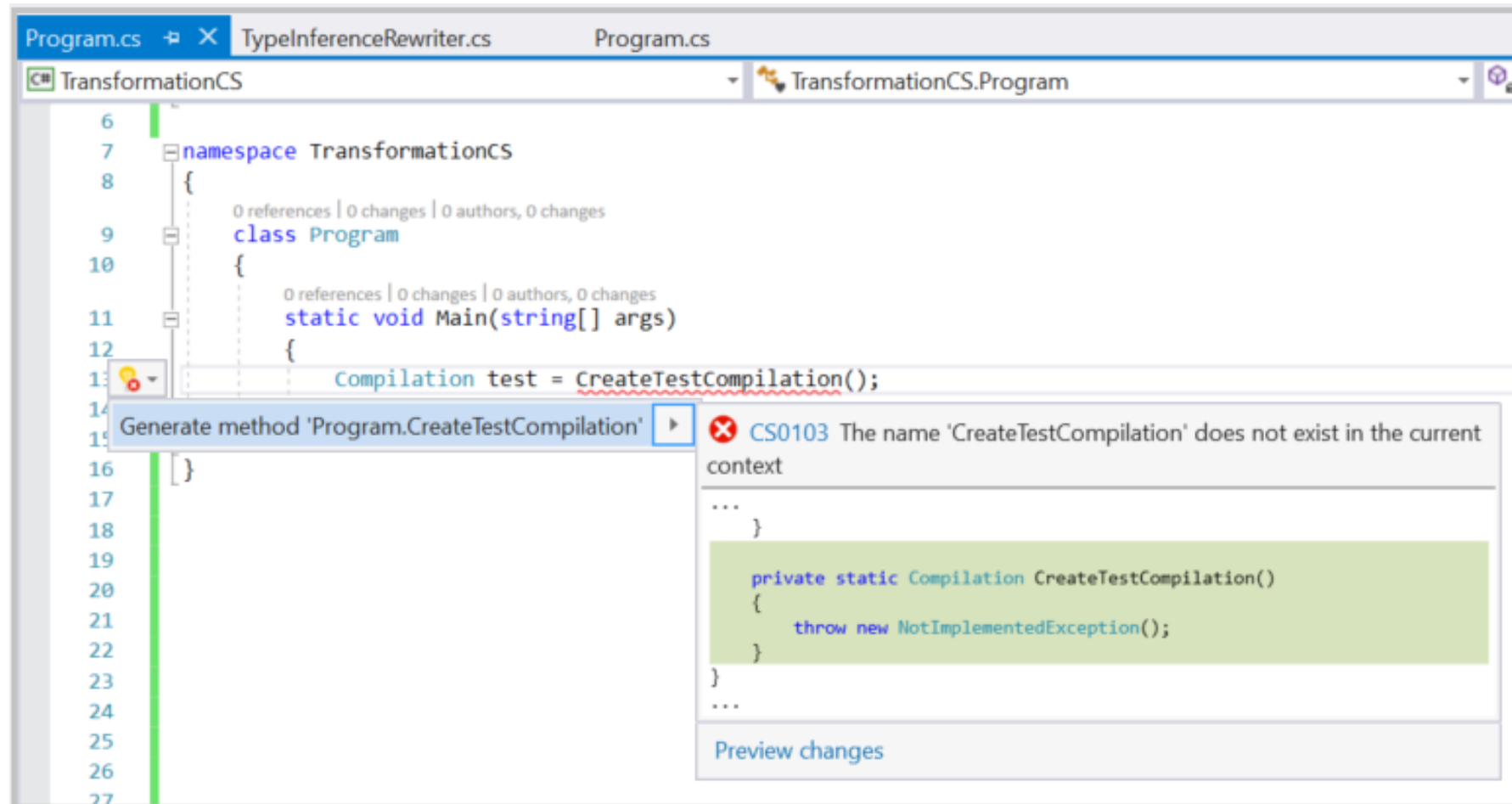
DEMO: SCRIPTING API

```
PS C:\Users\Ali Bahraminezhad\source\repos\Scripter> |
```

I

<https://github.com/dotnet/roslyn/blob/main/docs/wiki/Scripting-API-Samples.md>

DEMO: EDITOR EXTENSION (DIAGNOSTIC ANALYZER, CODE FIX PROVIDER)





THANK YOU FOR YOUR ATTENTION