



# MODELLALAPÚ SZOFTVERFEJLESZTÉS

## IV. GYAKORLAT COMPILER AS A SERVICE

DR MEZEI GERGELY

## BLACK BOX – WHITE BOX

- Hogyan működik egy szoftverfejlesztő eszköz? (pl. IntelliJ)
  - Szerkesztjük a kódot
    - Syntax highlight
    - Szintaktikai hibák
    - Kódkiegészítés
  - Futtatjuk a kódot

A fordítás folyamata egy fekete doboz

# BLACK BOX – WHITE BOX

- Compiler, mint fekete doboz
  - Mi történik ha hibás? (syntax highlight)
  - Hogyan terjeszthetjük ki?
  - Sok esetben elég, de...
    - A fejlesztőeszköz nem lát bele a fordítóba
    - AI-vezérelt kódkiegészítés? (pl. Github Copilot)

# BLACK BOX – WHITE BOX

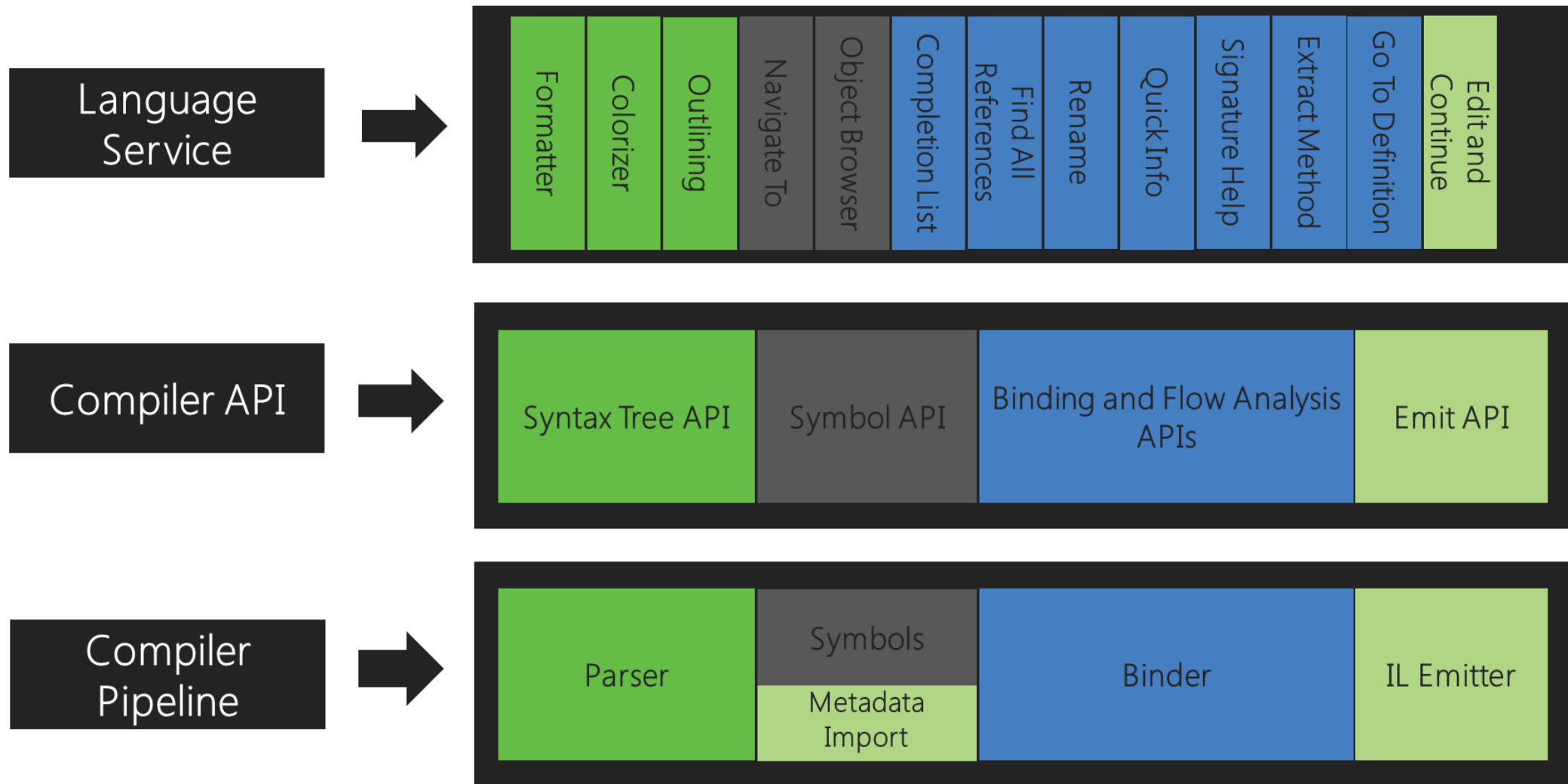
- Compiler – fehér dobozként?
- Roslyn – Compiler as a Service
  - Kódanalízálás
  - Kódfixálás
  - Refaktorálás
  - Kódgenerálás
  - Visual Studioba beépülve

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

# COMPILER AS A SERVICE

- CaaS – miért jó?
  - Saját (csapatszintű) kódkonvenciók követése
  - Könyvtárak javasolt használata
  - Segítség az általános konvenciókhoz, mintákhoz

# ROSLYN - FELÉPÍTÉS



# ROSLYN - FELÉPÍTÉS

- Compiler APIs
  - A fordítás lépései során előálló információk
- Diagnostic APIs
  - Kód diagnosztika, MSBuild és VS integráció
- Scripting APIs
  - Kódvégrehajtás
- Workspace APIs
  - Solution-szintű analízis és refaktorálás

# DEMO: SZINTAKTIKAI ELEMZÉS (SYNTAX ANALYZER)

The screenshot displays the Syntax Visualizer application. On the left, the 'Syntax Tree' panel shows a hierarchical view of the code structure, including 'UsingDirective', 'NamespaceDeclaration', 'ClassDeclaration', and 'MethodDeclaration'. The 'Properties' panel below it shows details for the selected 'MethodDeclarationSyntax' node, such as 'Type', 'Kind', 'AttributeLists', and 'Body'. The main editor shows the source code for 'Program.cs' with a highlighted method: 

```
static void Main(string[] args) { Console.WriteLine("Hello World!"); }
```

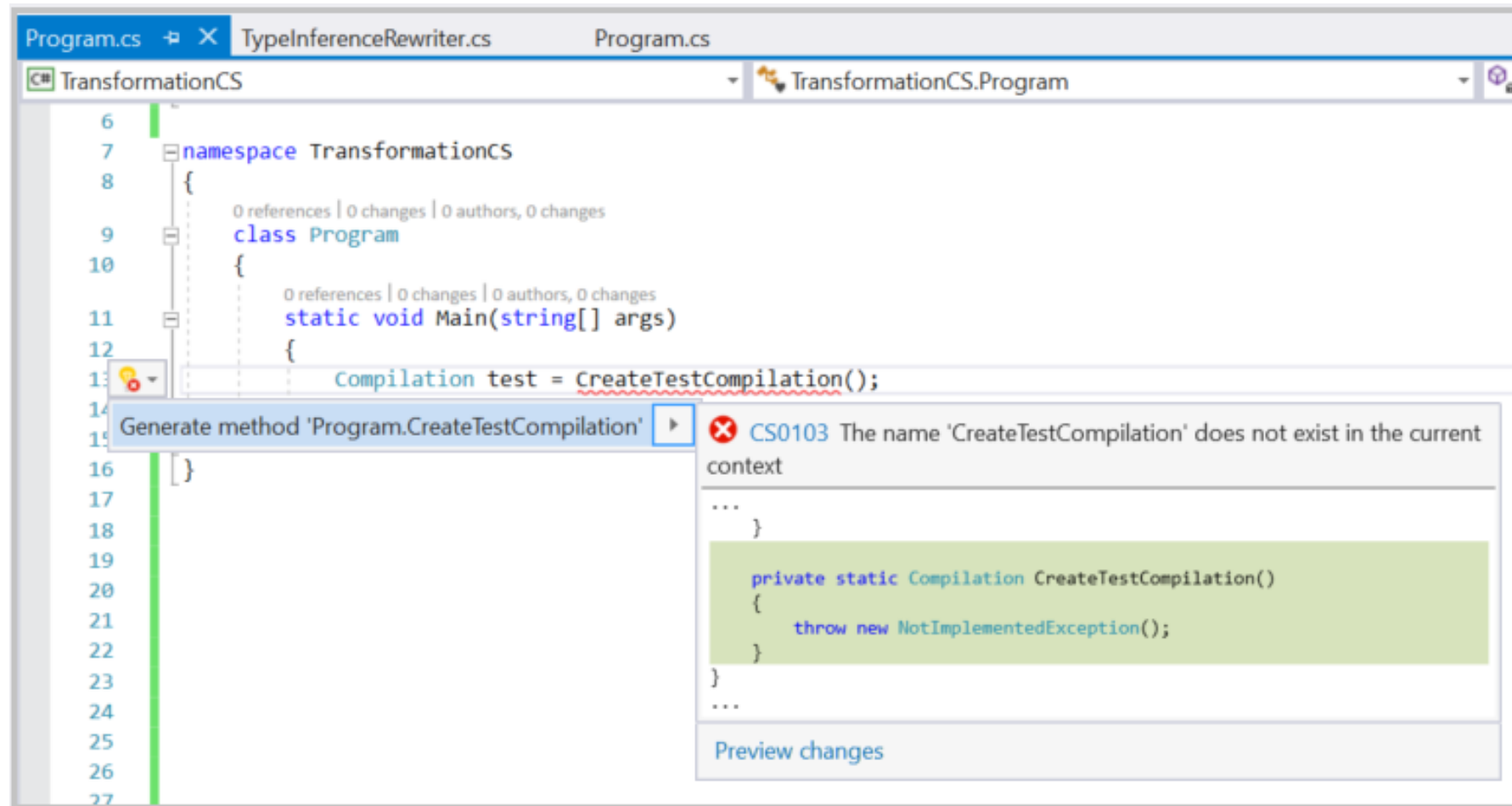
. Below the code editor, the 'Syntax.dgml' panel displays a detailed syntax tree diagram for the selected method. The tree is rooted at 'MethodDeclaration Node' and branches out to include nodes for 'Static Token', 'Void Token', 'ParameterList Node', 'Block Node', 'ExpressionStatement Node', 'InvocationExpression Node', 'ArgumentList Node', and various tokens like 'string', 'Console', and 'WriteLine'.



## DEMO: SZEMANTIKAI ELEMZÉS (DATA FLOW, CONTROL FLOW)

- Felhasználhatóak a szemantikai elemzés eredményei
  - Adatfolyam
    - Milyen szimbólumok vannak definiálva?
    - Melyik szimbólum hol érhető el? Mit fed az adott név?
    - Használja-e bárki az adott szimbólumot (változót)?
  - Vezérlésfolyam
    - Milyen értékekkel térhet vissza egy adott metódus?

# DEMO: EDITOR BŐVÍTÉS (DIAGNOSTIC ANALYZER, CODE FIX PROVIDER)



# DEMO: FUTÁSIDEJŰ KÓDFUTTATÁS (SCRIPTING API)

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

I

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



KÖSZÖNJÜK A FIGYELMET!