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

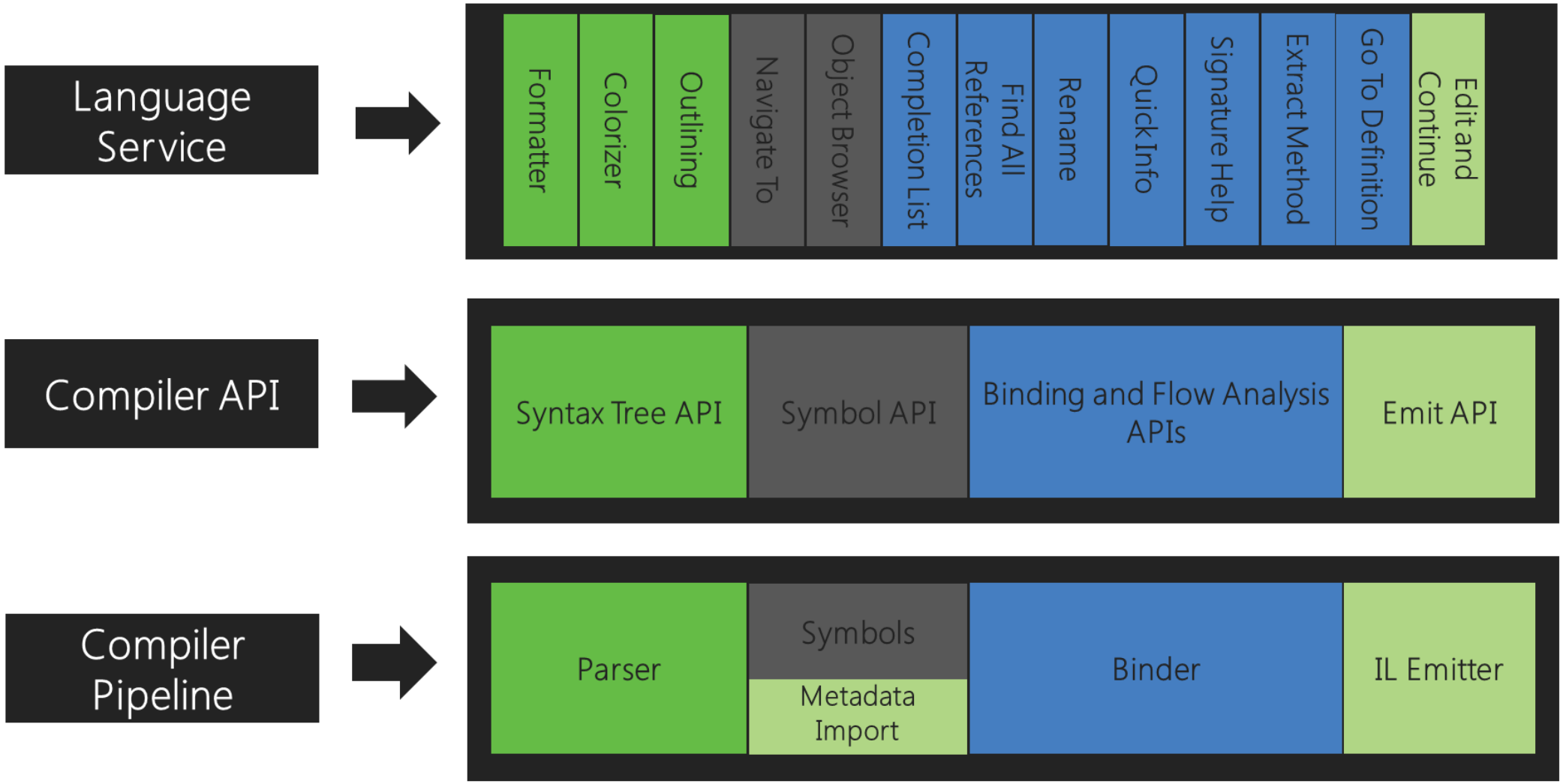
- Compiler – as a white box?
- Roslyn – Compiler as a Service
  - Code analyzer
  - Code fixer
  - Refactoring
  - Code generation
  - Built into Visual Studio

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

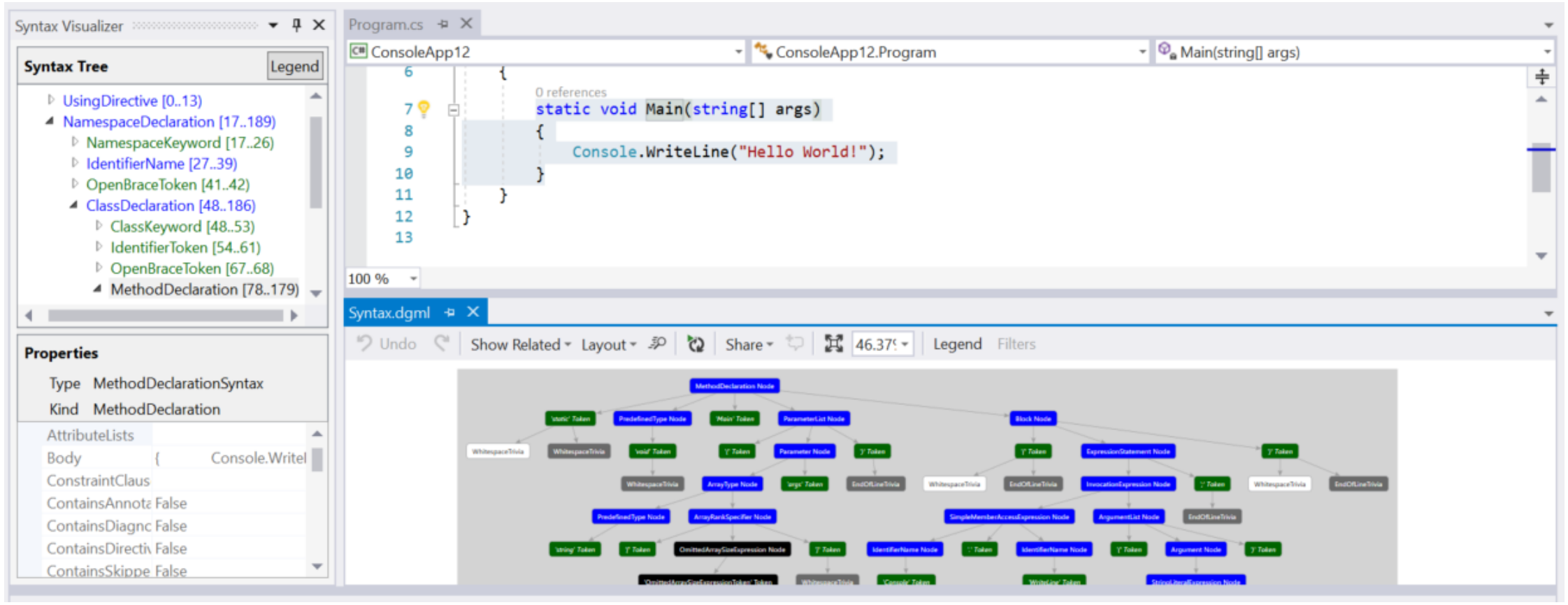
COMPILER AS A SERVICE

- 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



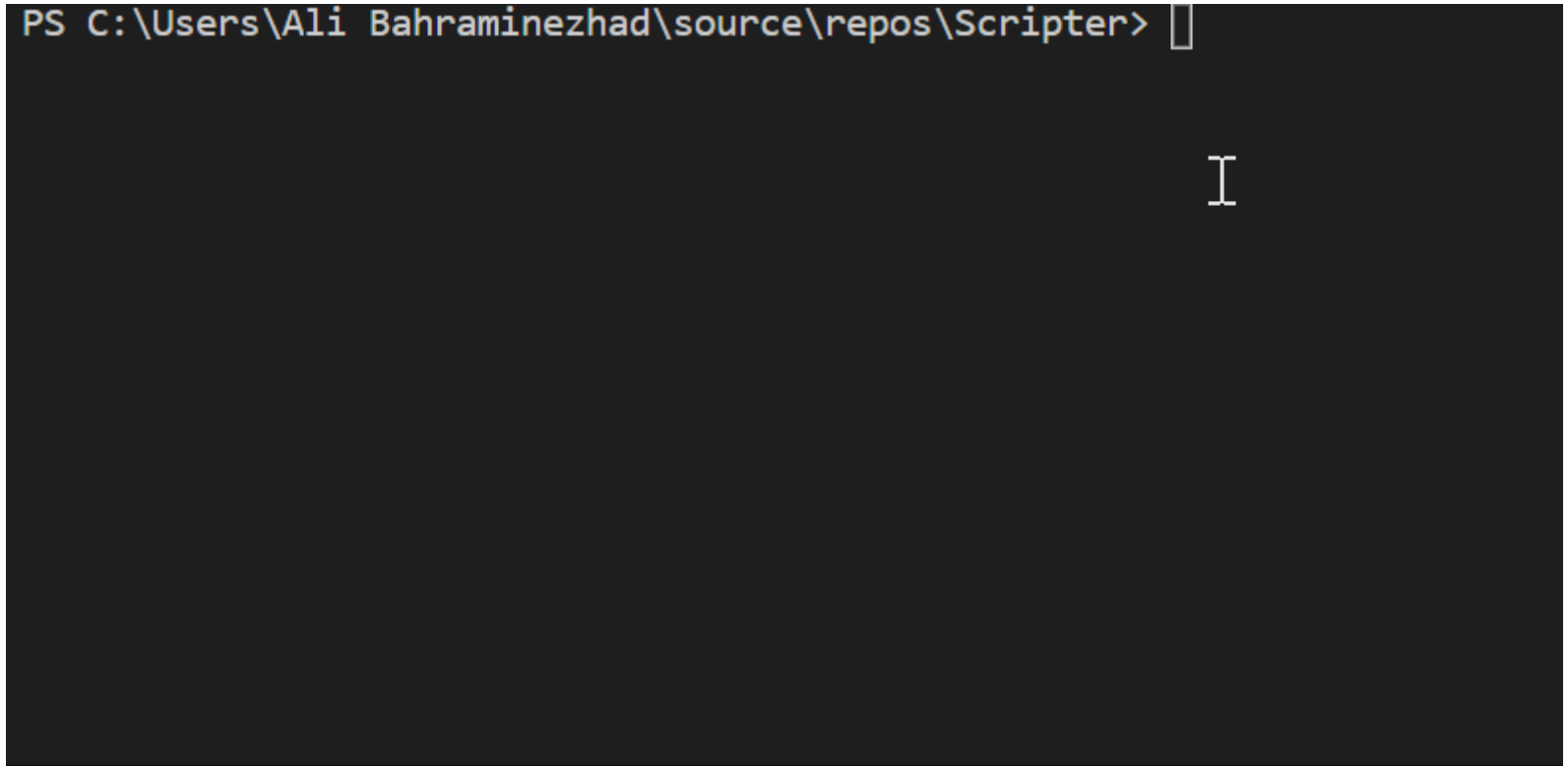
DEMO: SYNTAX ANALYZER



DEMO: DATA FLOW, CONTROL FLOW

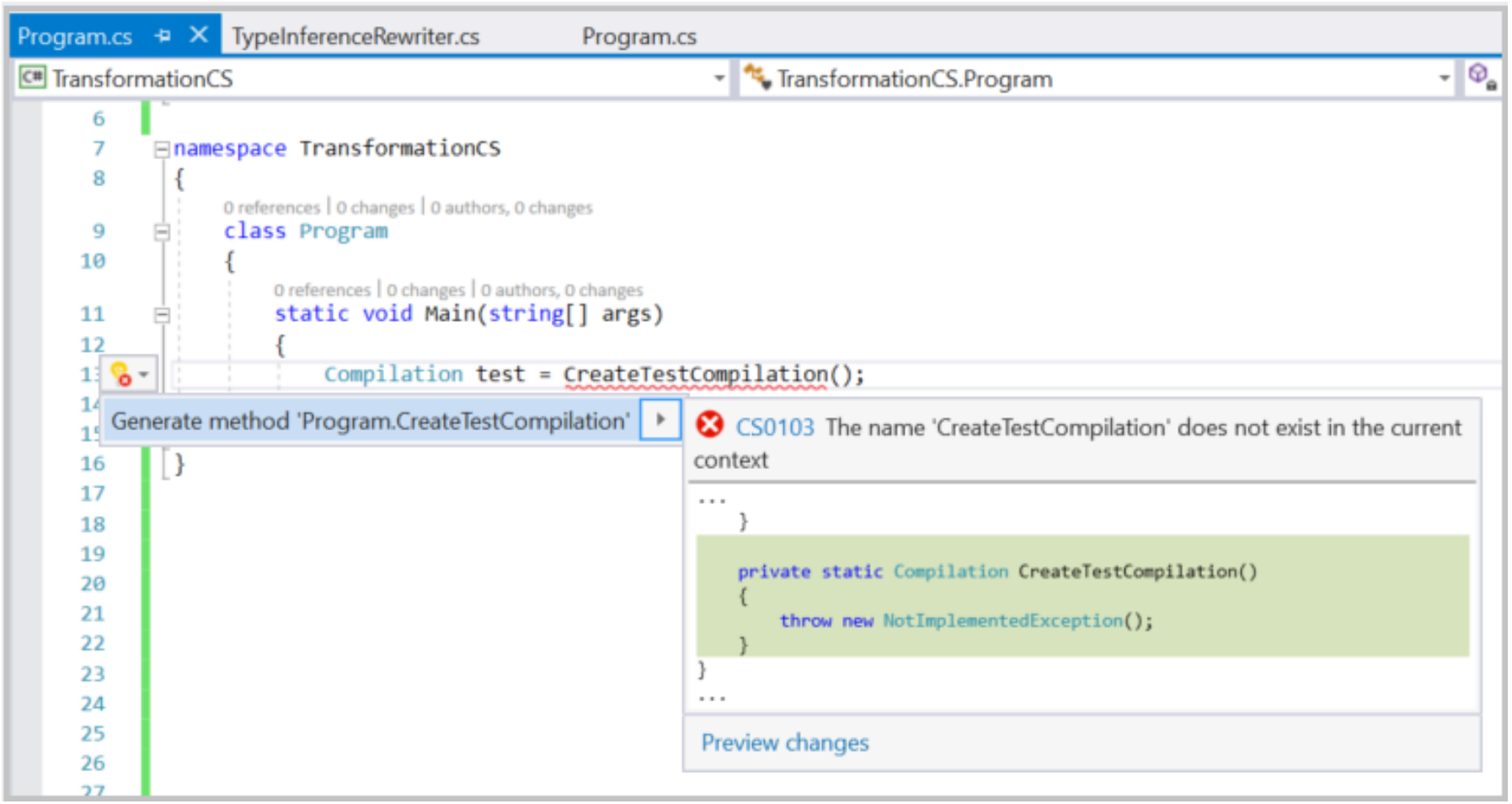
- Artifacts of the semantic analysis can be used
  - Data flow
    - What symbols are defined?
    - Which symbols are available? What are their scopes? Name resolution
    - Is the given symbol (variable) used at all?
  - Control flow
    - Possible return values (not type!) of the methods

DEMO: SCRIPTING API



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