W drodze ku... Roslyn

```
using System;
namespace HelloWorld {
class Hello {
   static void Main()
   { Console.WriteLine("Hello World!");
   }
  }
}
```

```
using System;
namespace HelloWorld {
class Hello {
   static void Main()
   { Console.WriteLine("Hello World!");
   }
  }
}
```

F5

```
using System;
namespace HelloWorld {
class Hello {
   static void Main()
   { Console.WriteLine("Hello World!");
   }
  }
}
```

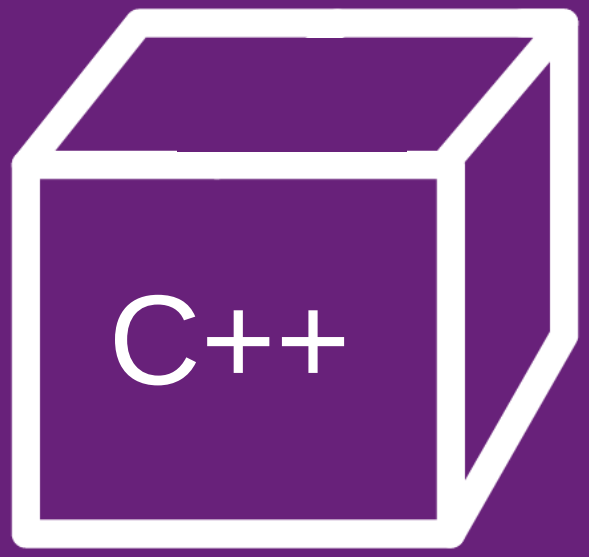
F5

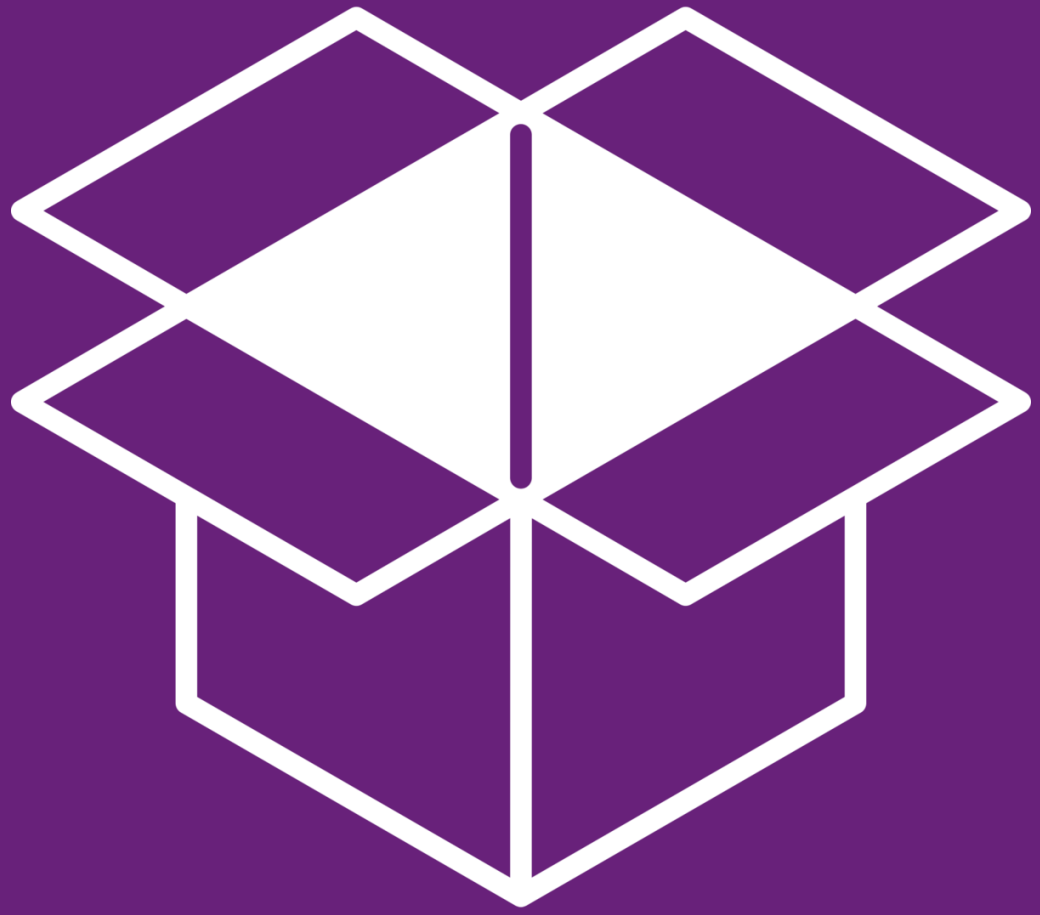
.exe
.dll

F5 ???


MAGIC

Kompilator


C++



Roslyn

Visual Studio 2015

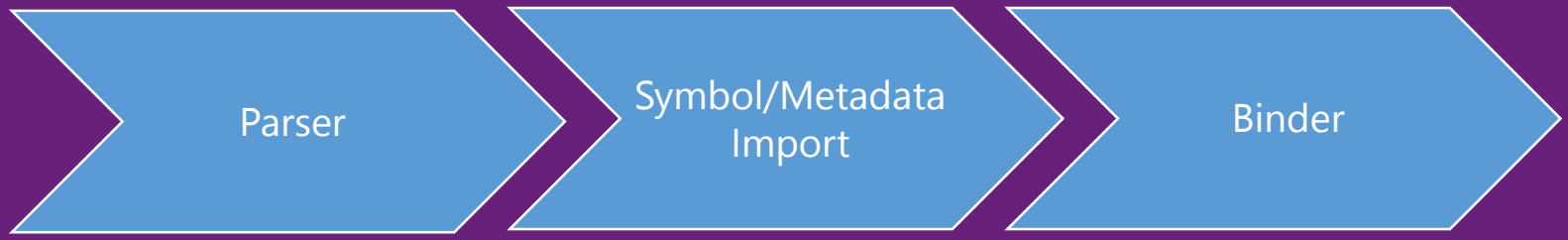
Jak zacząć ?

Kompilacja

Proces kompilacji

Proces kompilacji

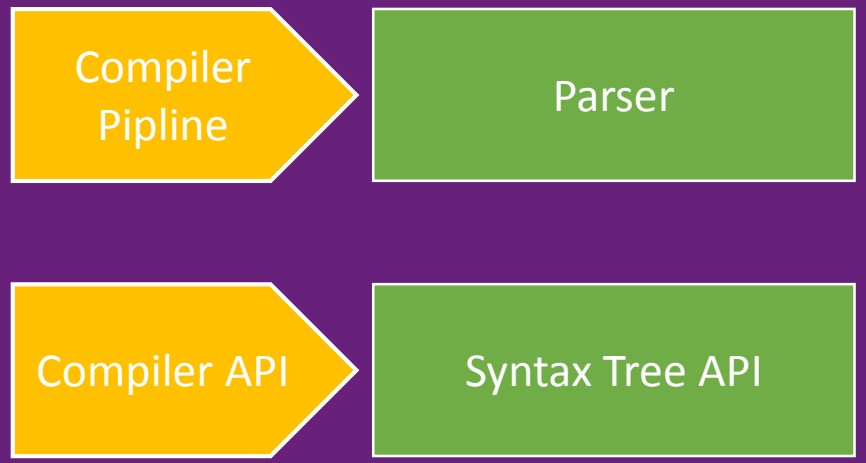
Proces kompilacji

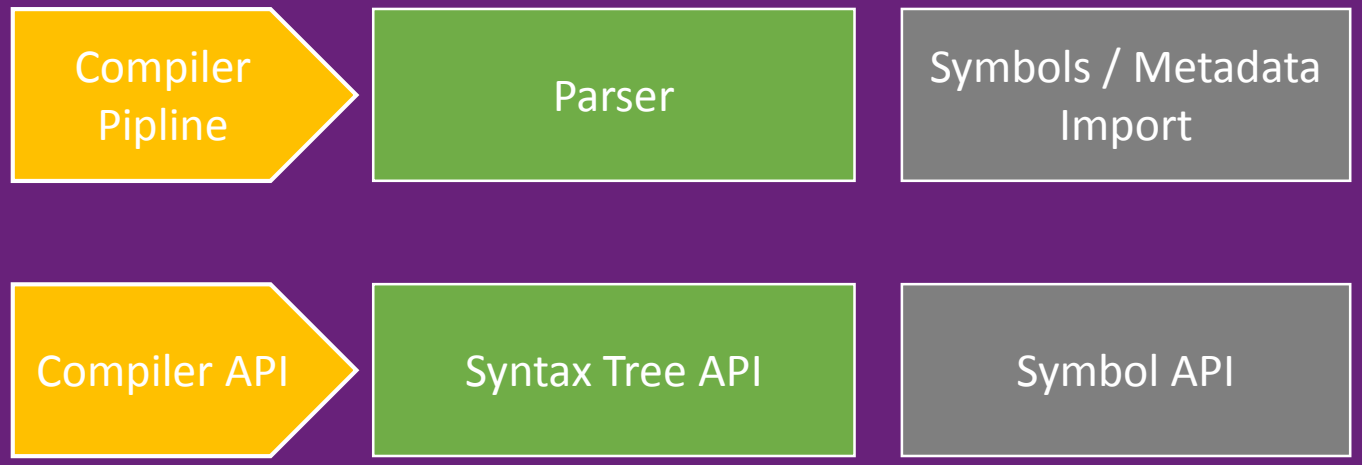


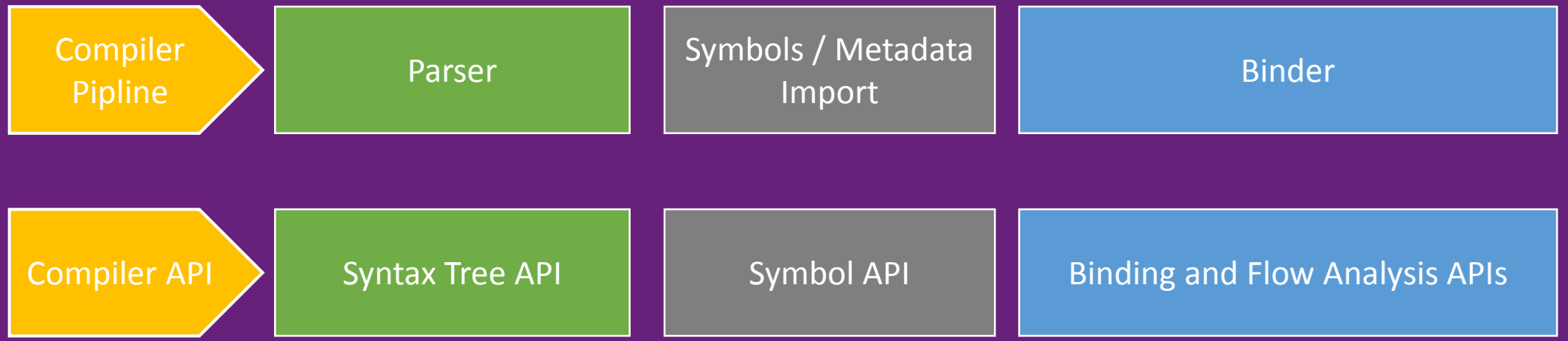
Proces kompilacji

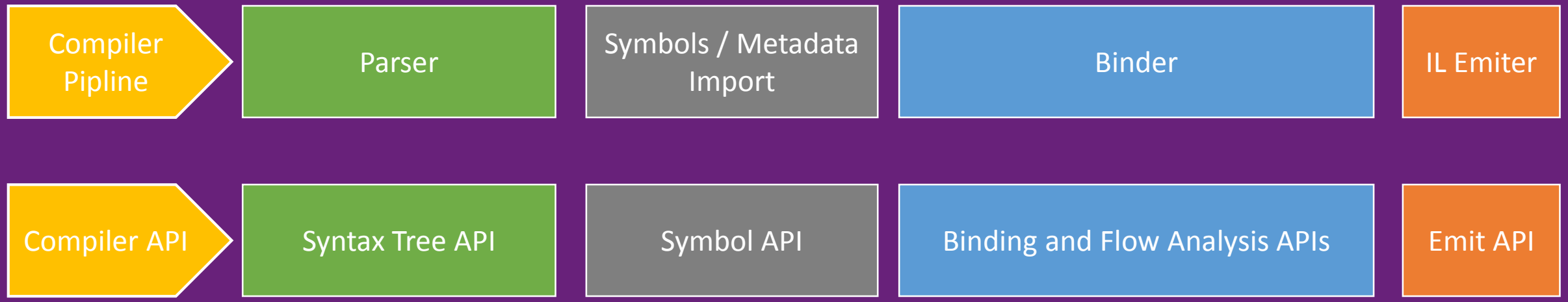
Compiler API


Compiler Pipline
Parser
Symbols / Metadata Import
Binder
IL Emiter


Compiler
Pipline
Parser
Compiler API
Syntax Tree API

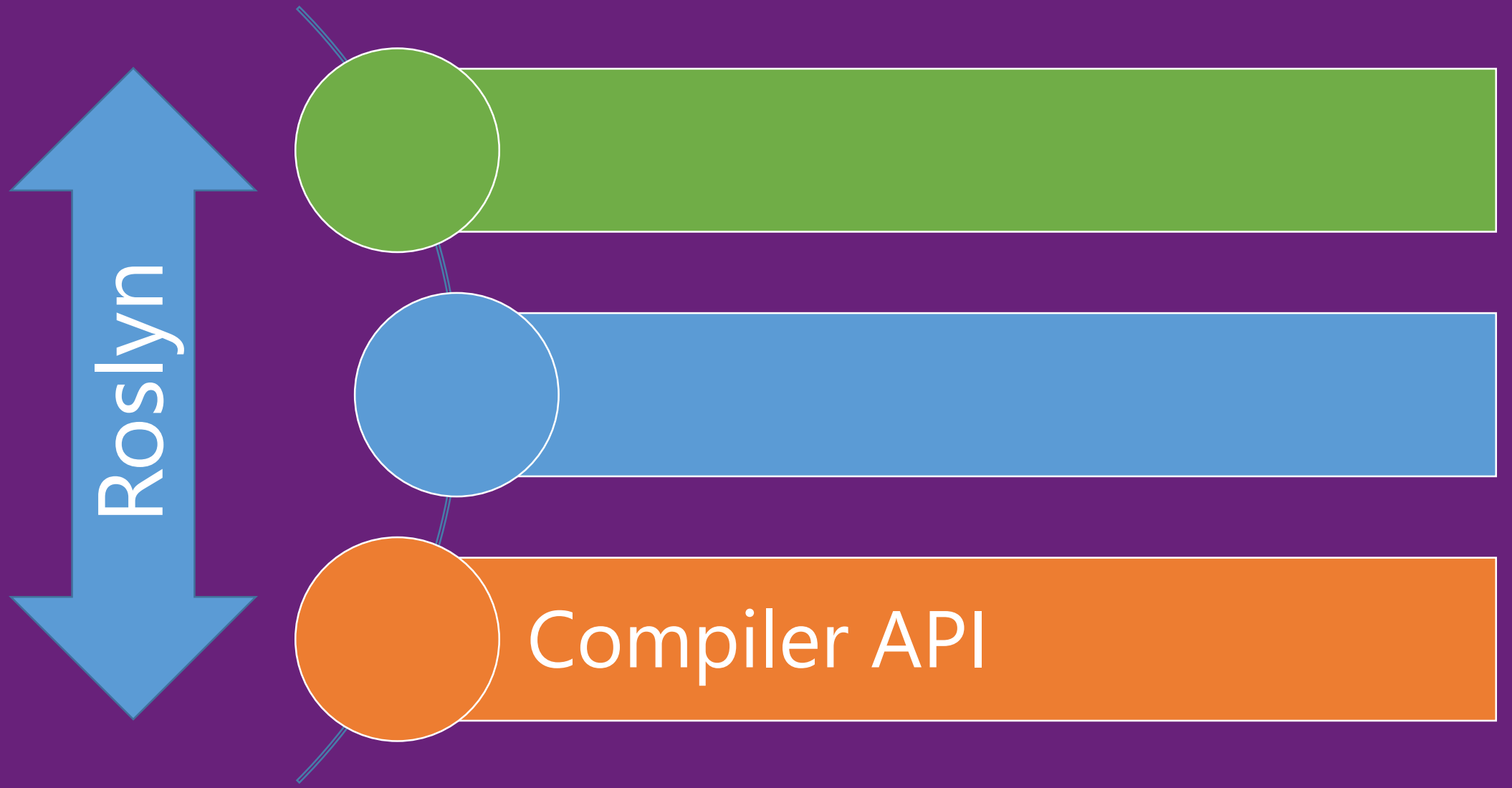

Compiler Pipline
Parser
Symbols / Metadata Import
Compiler API
Syntax Tree API
Symbol API

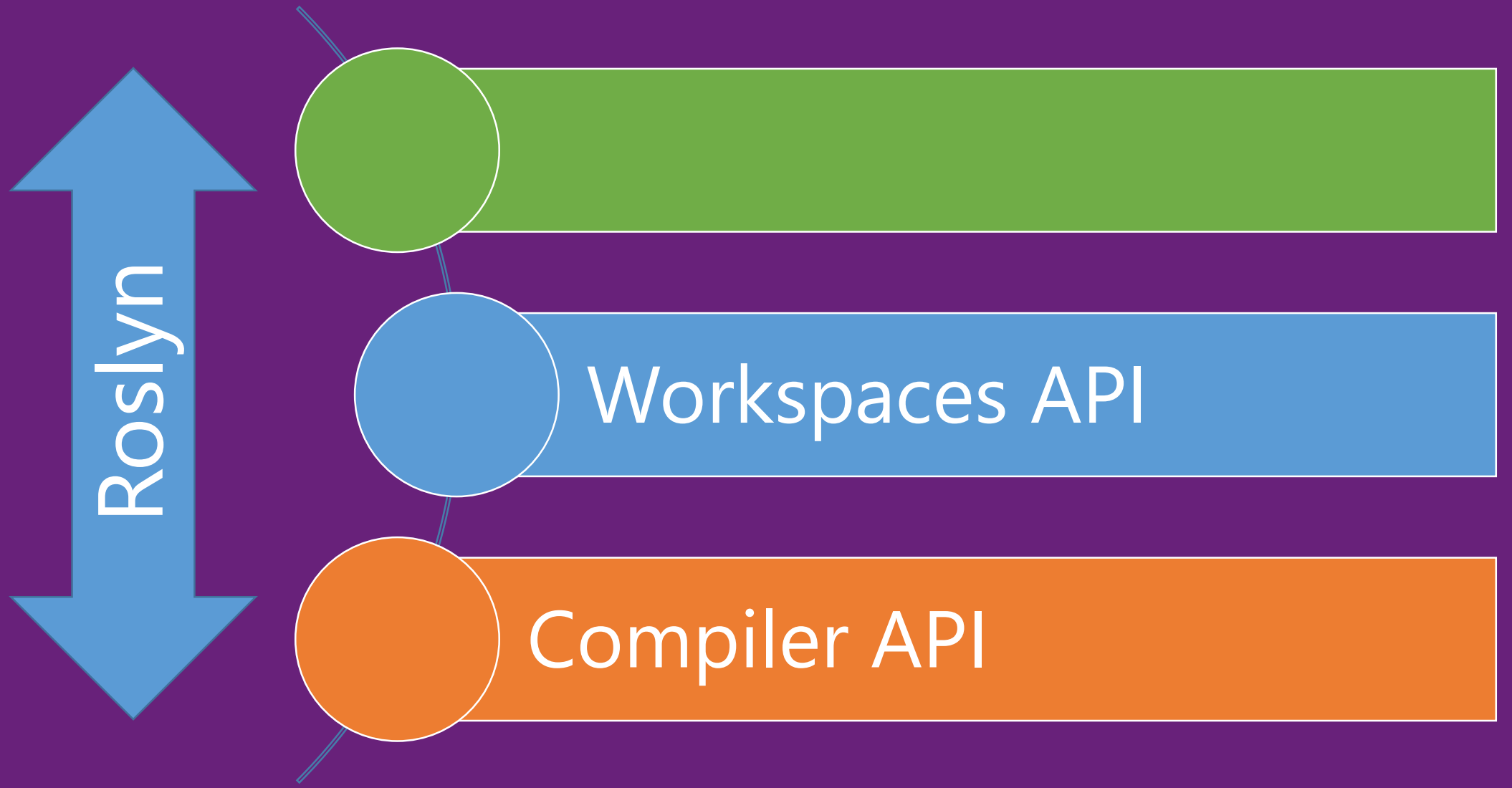

Compiler Pipline
Parser
Symbols / Metadata Import
Binder
Compiler API
Syntax Tree API
Symbol API
Binding and Flow Analysis APIs

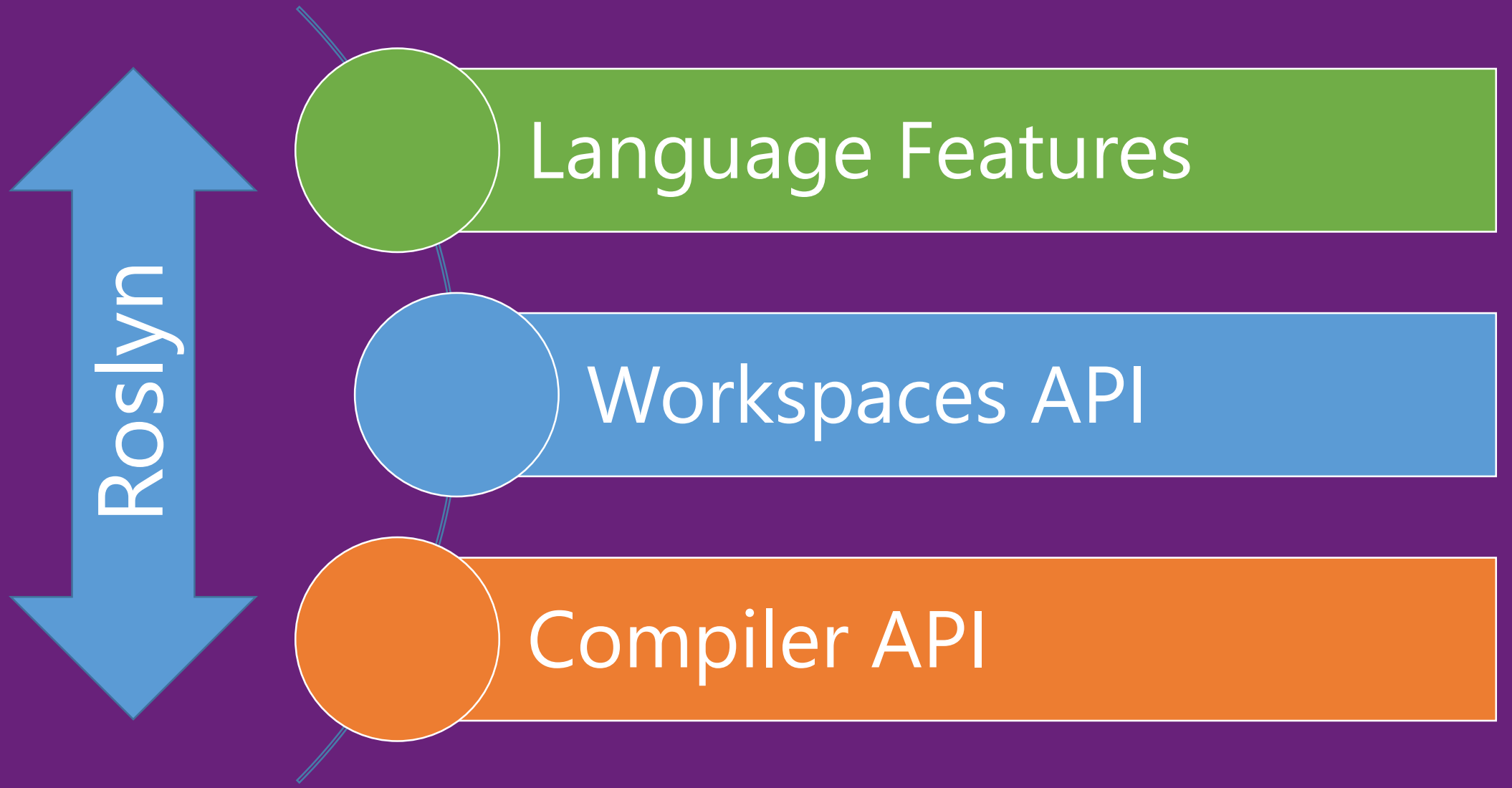

Compiler Pipline
Parser
Symbols / Metadata Import
Binder
IL Emiter
Compiler API
Syntax Tree API
Symbol API
Binding and Flow Analysis APIs
Emit API

Przykład 1 – Compiler API

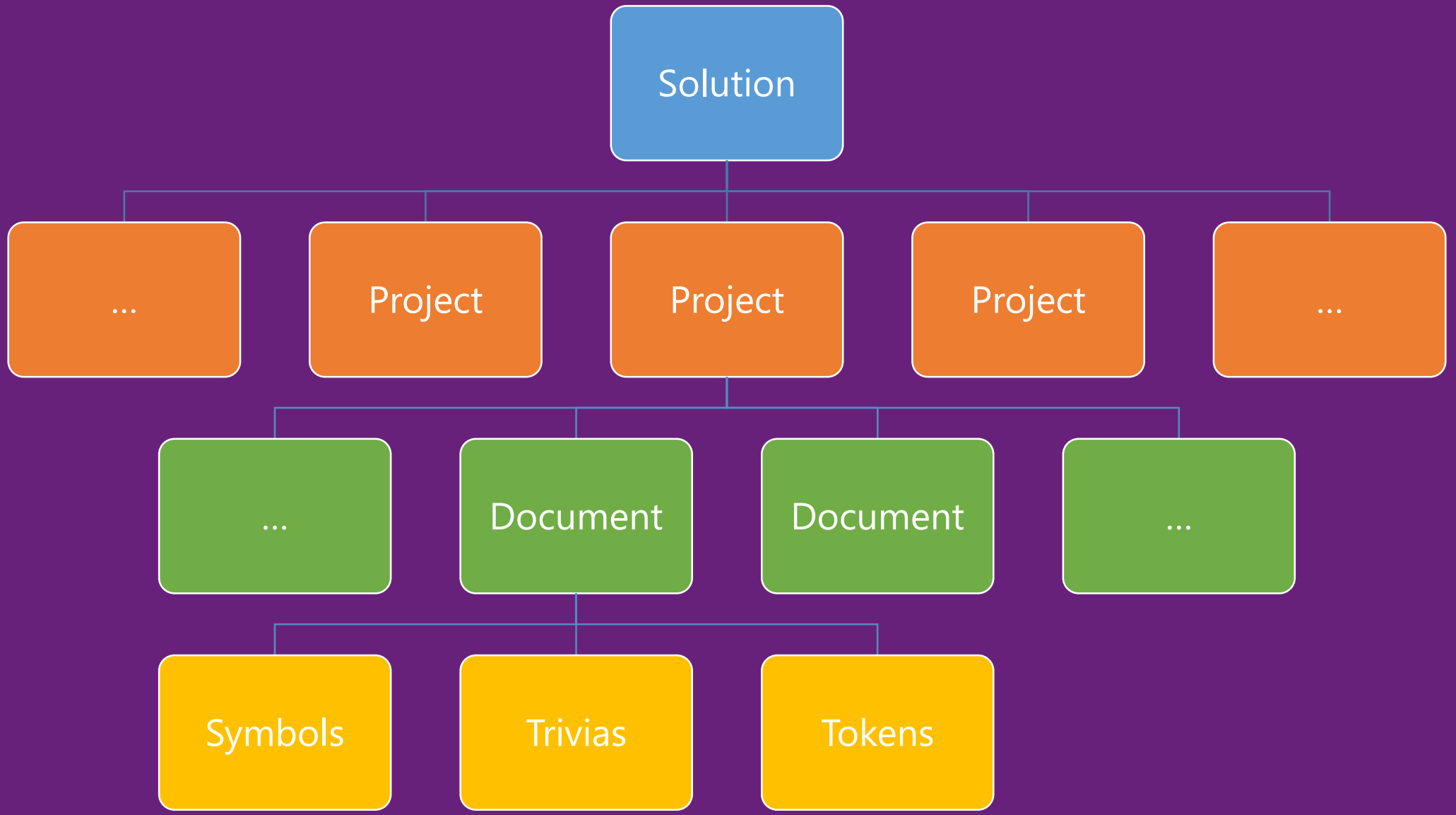

F*CK YEAH
IT WORKS
memegenerator.net

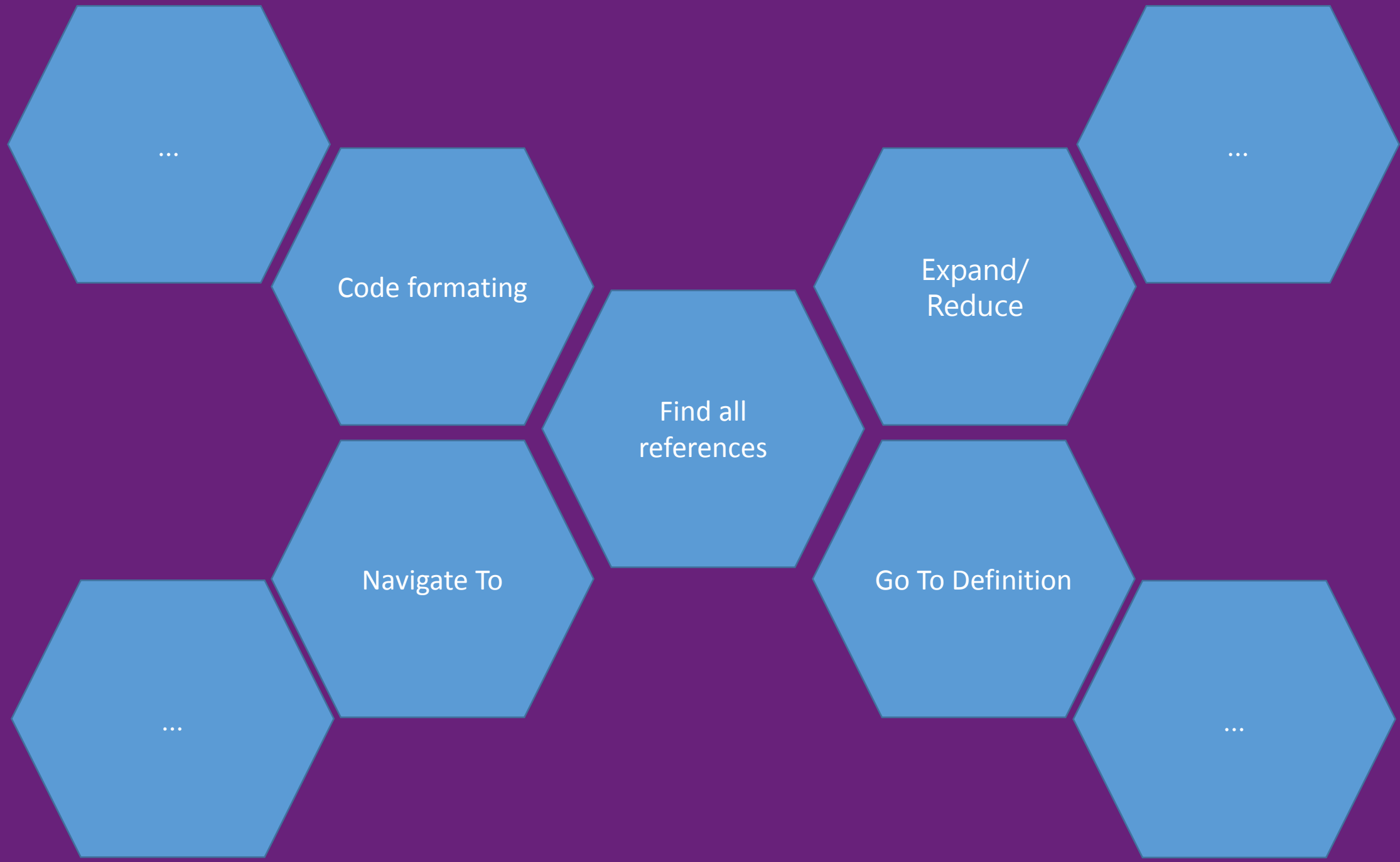

Roslyn
Compiler API


Roslyn
Workspaces API
Compiler API


Roslyn
Language Features
Workspaces API
Compiler API

Workspaces API


Solution
...
Project
Project
Project
...
...
Document
Document
...
Symbols
Trivias
Tokens


...
Code formating
Find all references
Expand/ Reduce
...
Navigate To
Go To Definition
...
...

Features API

- Code refactorings
- Code fixes

Przykład 2 – Features API

Przykład 3 – Features API

Language Features

C# 6.0

Auto inicjalizacja właściwości

```
public class Customer
{
 public string First { get; set; } = "Jane";
 public string Last { get; set; } = "Doe";
}
```

Właściwości o tylko jednym akcesorze

```
public class Customer
{
 public string First { get; } = "Jane";
 public string Last { get; } = "Doe";
}
```

Definiowanie nowych metod za pomocą wyrażeń lambda

```
public Point Move(int dx, int dy) => new Point(x + dx, y + dy);

public static Complex operator +(Complex a, Complex b) => a.Add(b);

public static implicit operator string(Person p) => p.First + " " + p.Last;

public void Print() => Console.WriteLine(First + " " + Last);
```

Definiowanie nowych właściwości za pomocą wyrażeń lambda

```
public string Name => First + " " + Last;

public Customer this[long id] => store.LookupCustomer(id);
```

Korzystanie z klas statycznych

```
using System.Console;
using System.Math;
class Program
{
        static void Main()
        {
                WriteLine(Sqrt(3*3 + 4*4));
        }
}
```

Extension methods

```
using System.Linq.Enumerable; // Just the type, not the whole
namespace
class Program
{
 static void Main()
 {
 var range = Range(5, 17); // Ok: not extension
 var odd = Where(range, i => i % 2 == 1); // Error, not in scope
```

Null-conditional operators

```
int? length = customers?.Length; // null if customers is null
Customer first = customers?[0]; // null if customers is null

int length = customers?.Length ?? 0; // 0 if customers is null

int? first = (customers != null) ? customers[0].Orders.Count() : null;
int? first = customers?[0].Orders?.Count();
if (predicate?.Invoke(e) ?? false) { … }

PropertyChanged?.Invoke(this, args);
```

String interpolation

```
var s = String.Format("{0} is {1} year{{s}} old", p.Name, p.Age);

var s = "\{p.Name} is \{p.Age} year{s} old";

var s = "\{p.Name,20} is \{p.Age:D3} year{s} old";

var s = "\{p.Name} is \{p.Age} year\{(p.Age == 1 ? "" : "s")} old";

var s = $"{p.Name,20} is {p.Age:D3} year{{s}} old";
```

nameof expressions

```
(if x == null) throw new ArgumentNullException(nameof(x));

WriteLine(nameof(person.Address.ZipCode)); // prints "ZipCode"
```

Index initializers

```
var numbers = new Dictionary<int, string> {
 [7] = "seven",
 [9] = "nine",
 [13] = "thirteen"
};
```

Exception filters

```
try { … }
catch (MyException e) if (myfilter(e))
{
  …
}
```

Exception filters

```
try { … }
catch (MyException e) if (myfilter(e))
{
  …
}

private static bool Log(Exception e) { /* log it */ ; return false; }
…
try { … } catch (Exception e) if (Log(e)) {}
```

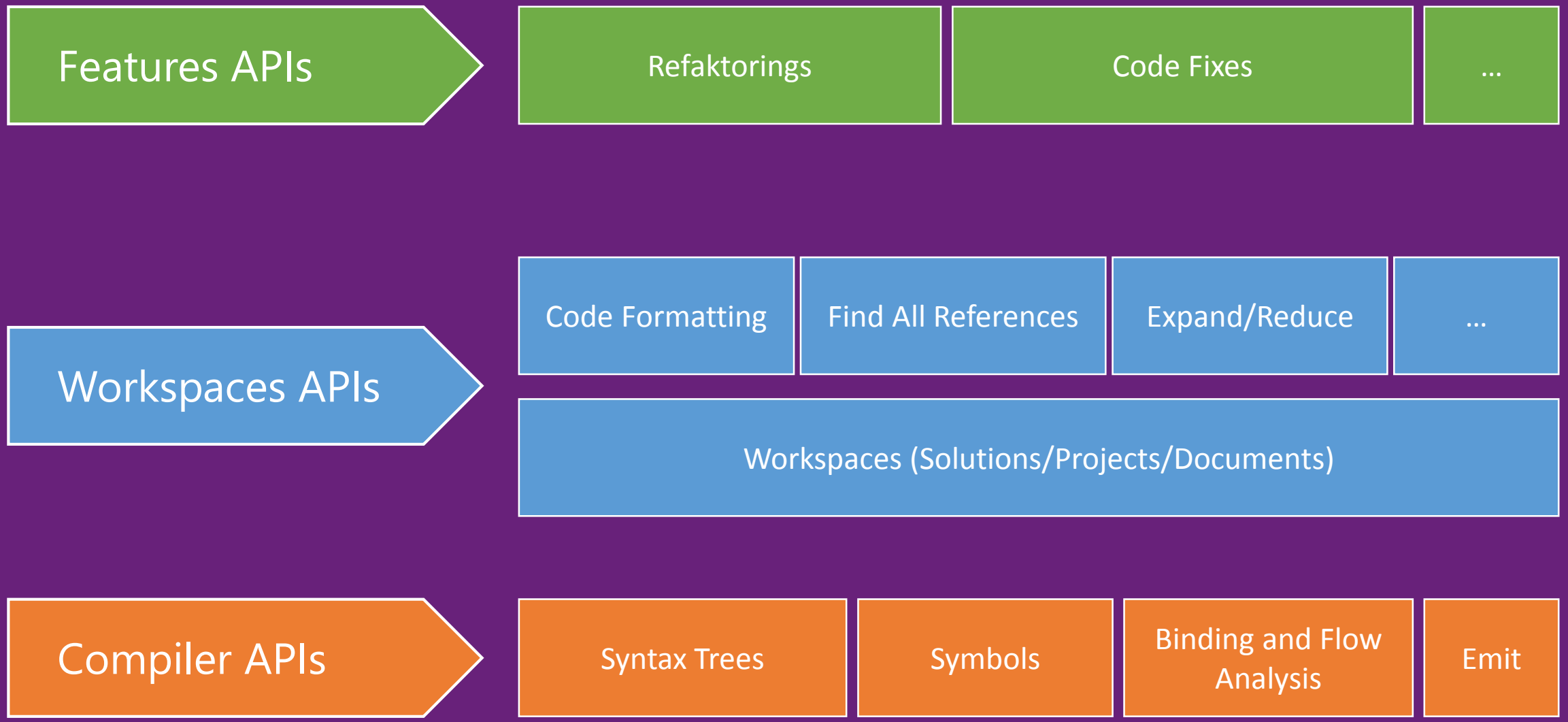
Await in catch and finally blocks

```
Resource res = null;
try
{
        res = await Resource.OpenAsync(…); // You could do this.
        …
}
catch(ResourceException e)
{
        await Resource.LogAsync(res, e); // Now you can do this …
}
finally
{
        if (res != null) await res.CloseAsync(); // … and this.
}
```

Parameterless constructors in structs

```
struct Person
{
 public string Name { get; }
 public int Age { get; }
 public Person(string name, int age) { Name = name; Age = age; }
 public Person() : this("Jane Doe", 37) { }
}
```

Podsumowanie


Features APIs
Refaktorings
Code Fixes
…
Workspaces APIs
Code Formatting
Find All References
Expand/Reduce
…
Workspaces (Solutions/Projects/Documents)
Compiler APIs
Syntax Trees
Symbols
Binding and Flow Analysis
Emit

Open Source

Continous Integration ++

Intelissense ≠ IDE

ReSharper ???

Nadal w budowie… !!!

Dzięki za uwagę ;)

smacznego  !!!!

marek.seweryn@comarch.com