Other Diagnostics

Kathleen Dollard

@kathleendollard | www.CodeRapid.com

Other Analyzer Methods

Language Version Specific

Refactorings

Analyzer Registration Methods

Scopes

Session

- Command line or batch - lifetime matches the batch compilation
- Hosted (like Visual Studio) - lifetime is whatever the host wants

Compilation

- Working unit, of a analysis; a single run of a set of analyzers

Code block

- Unit of executable code (example: method body or property accessor body)

Analyzer

- Instance of the class derived from Diagnostic Analyzer

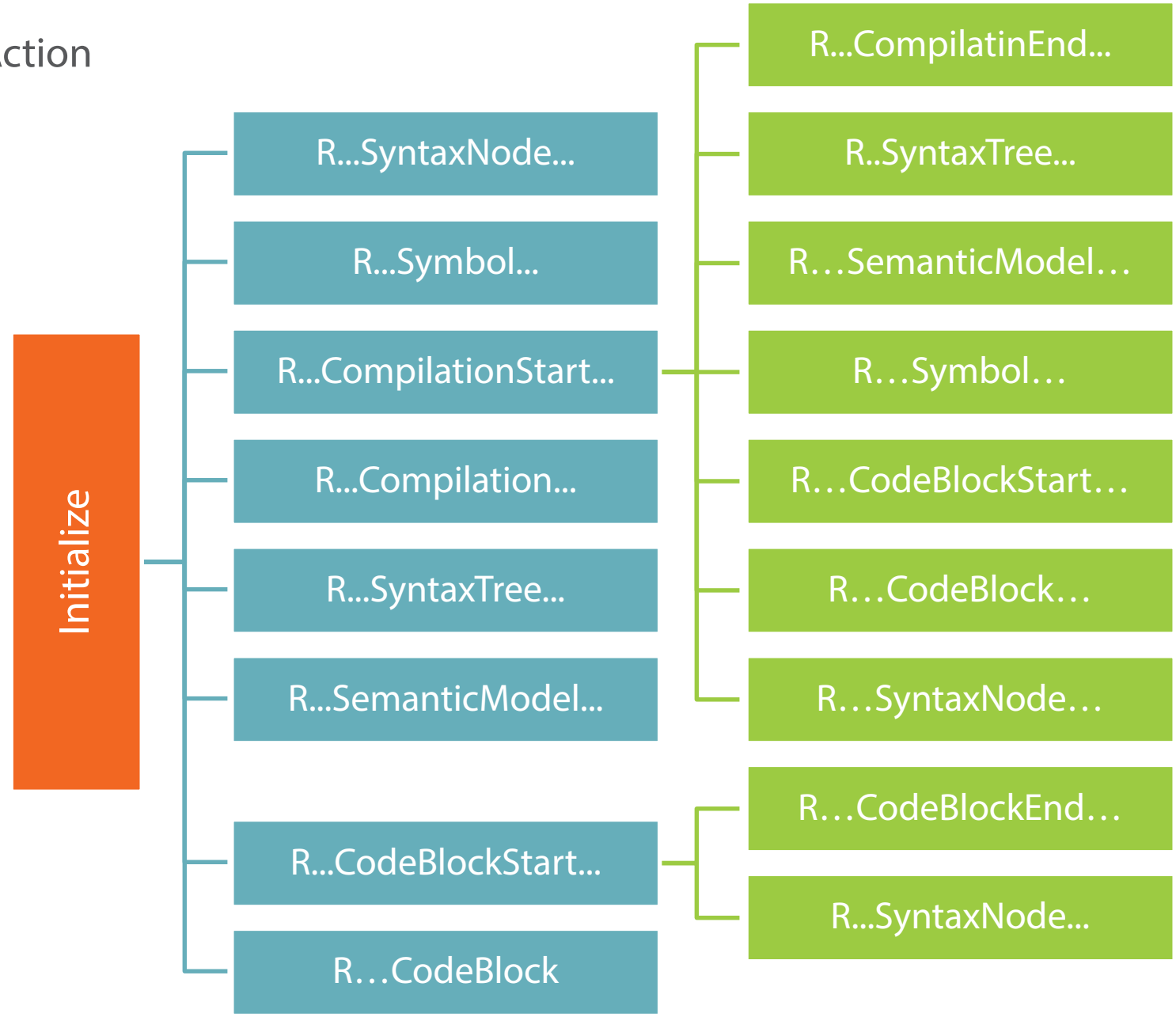
Action

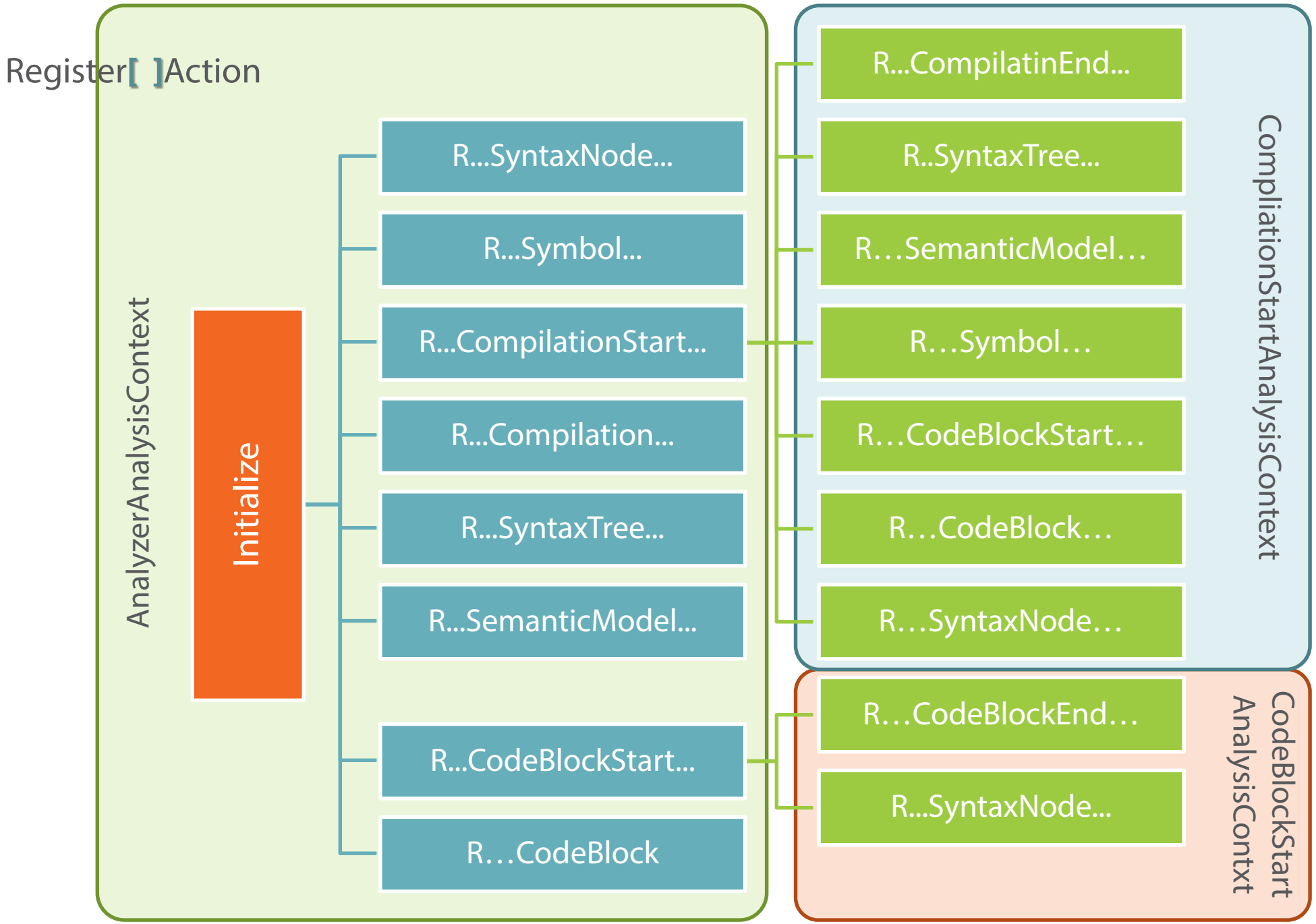
- A delegate registered with a Register…Action method of a context class

```
public override void Initialize(AnalysisContext context)
{
    context.RegisterSymbolAction(AnalyzeSymbol, SymbolKind.NamedType);
}
```

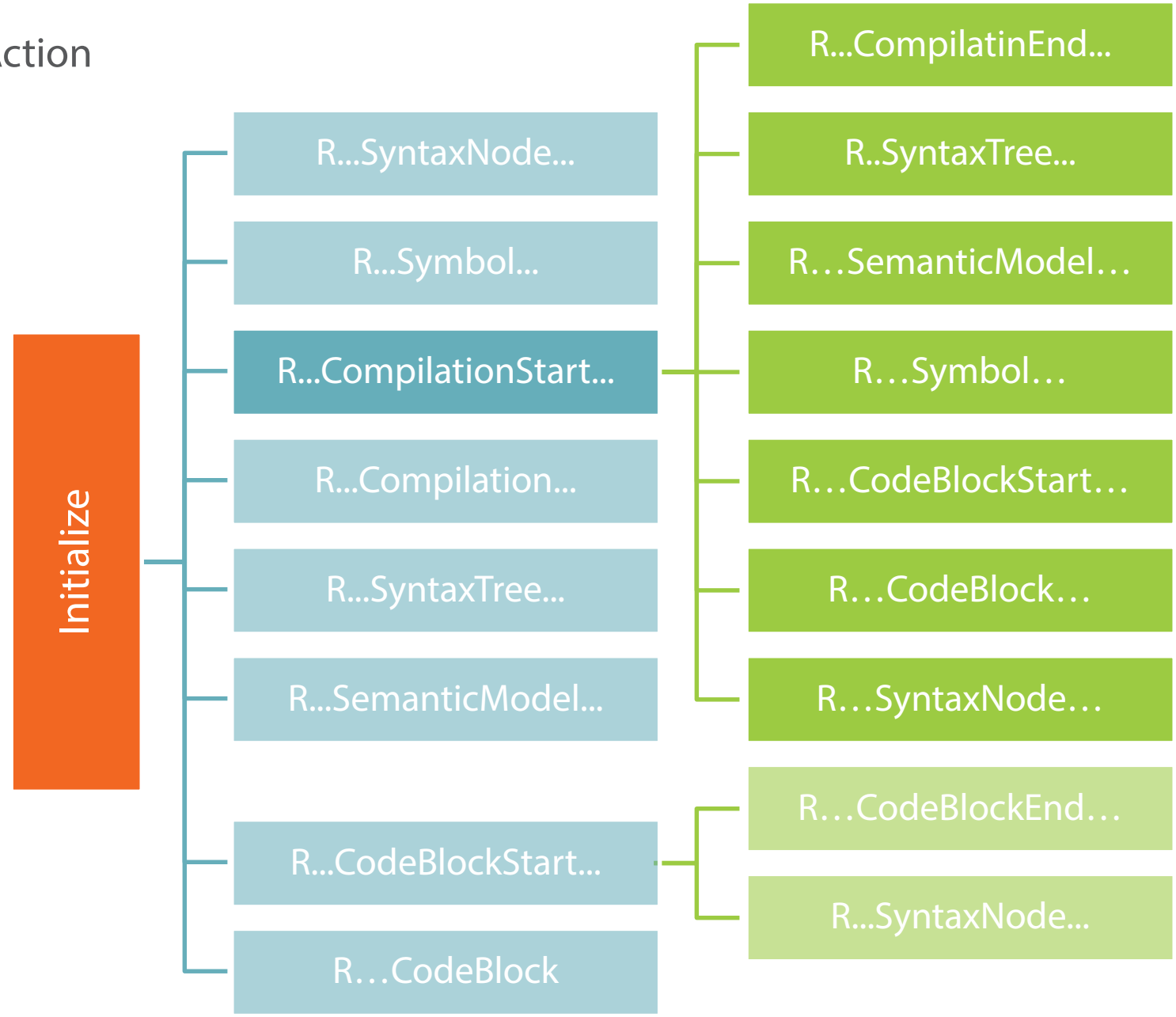
```
public override void Initialize(AnalysisContext context)
{
    int whatever = 42;
    context.RegisterSymbolAction(x=>AnalyzeSymbol(x, whatever),
                    SymbolKind.NamedType);
}
```

Register[ ]Action

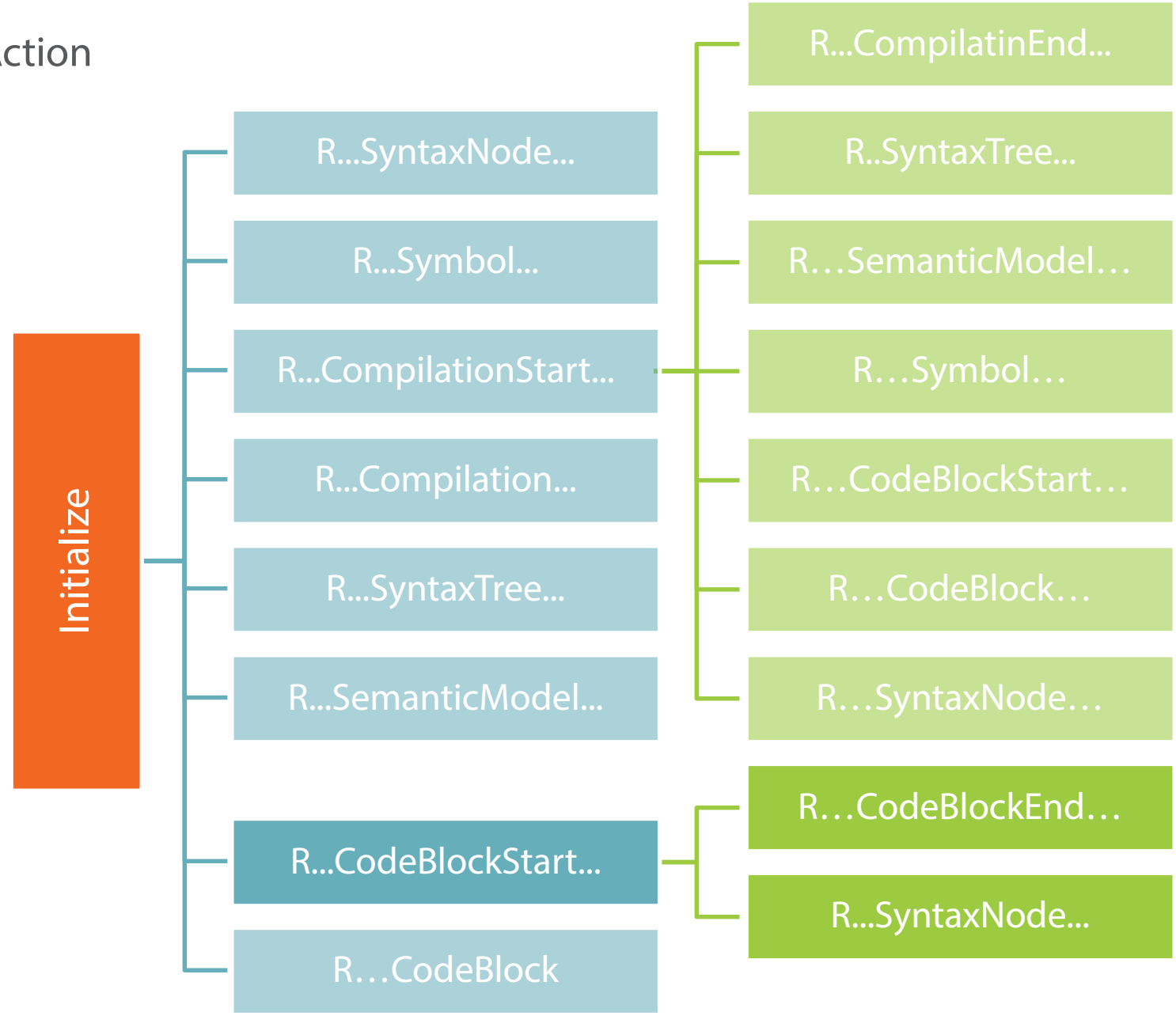




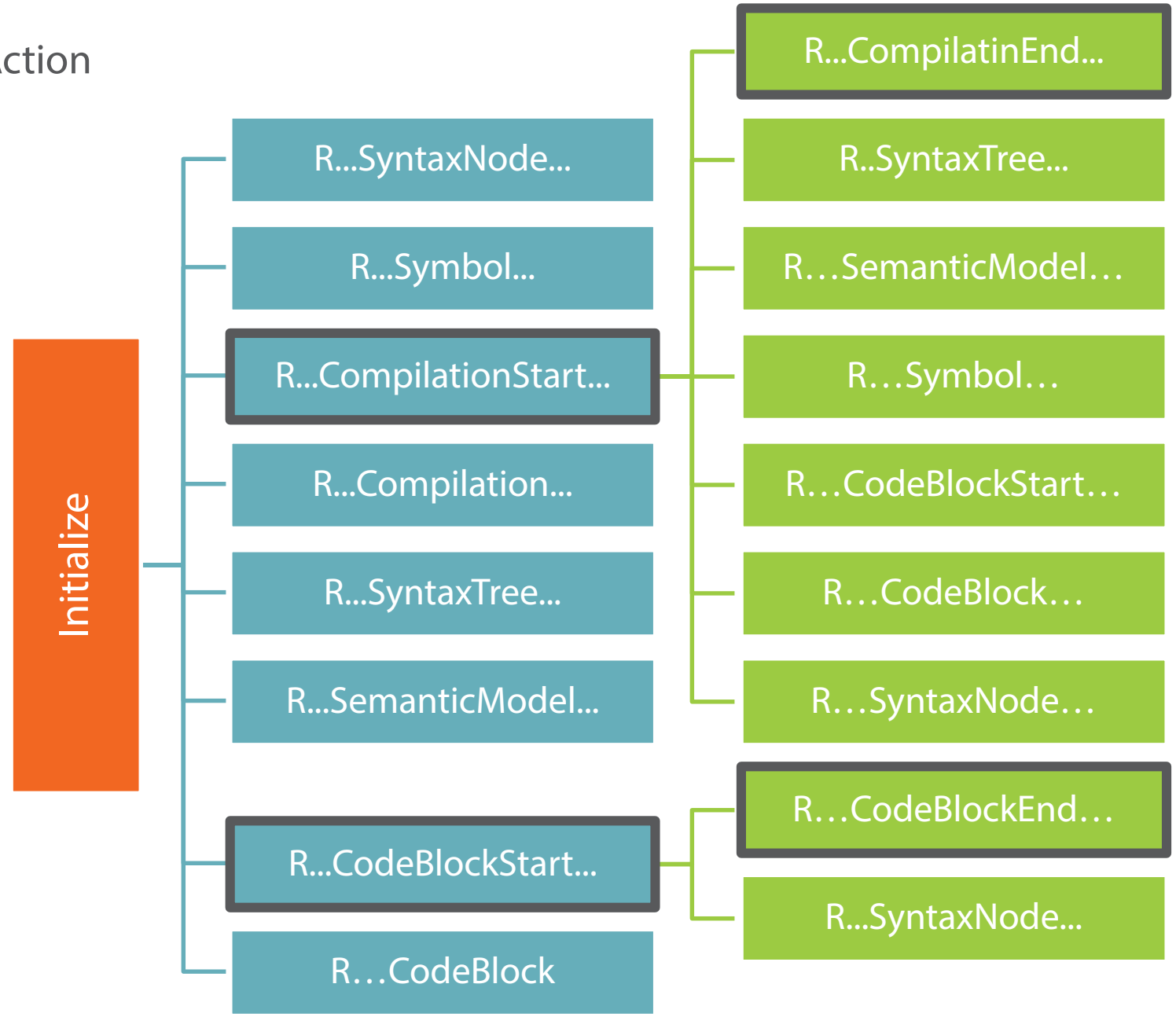
Register[ ]Action



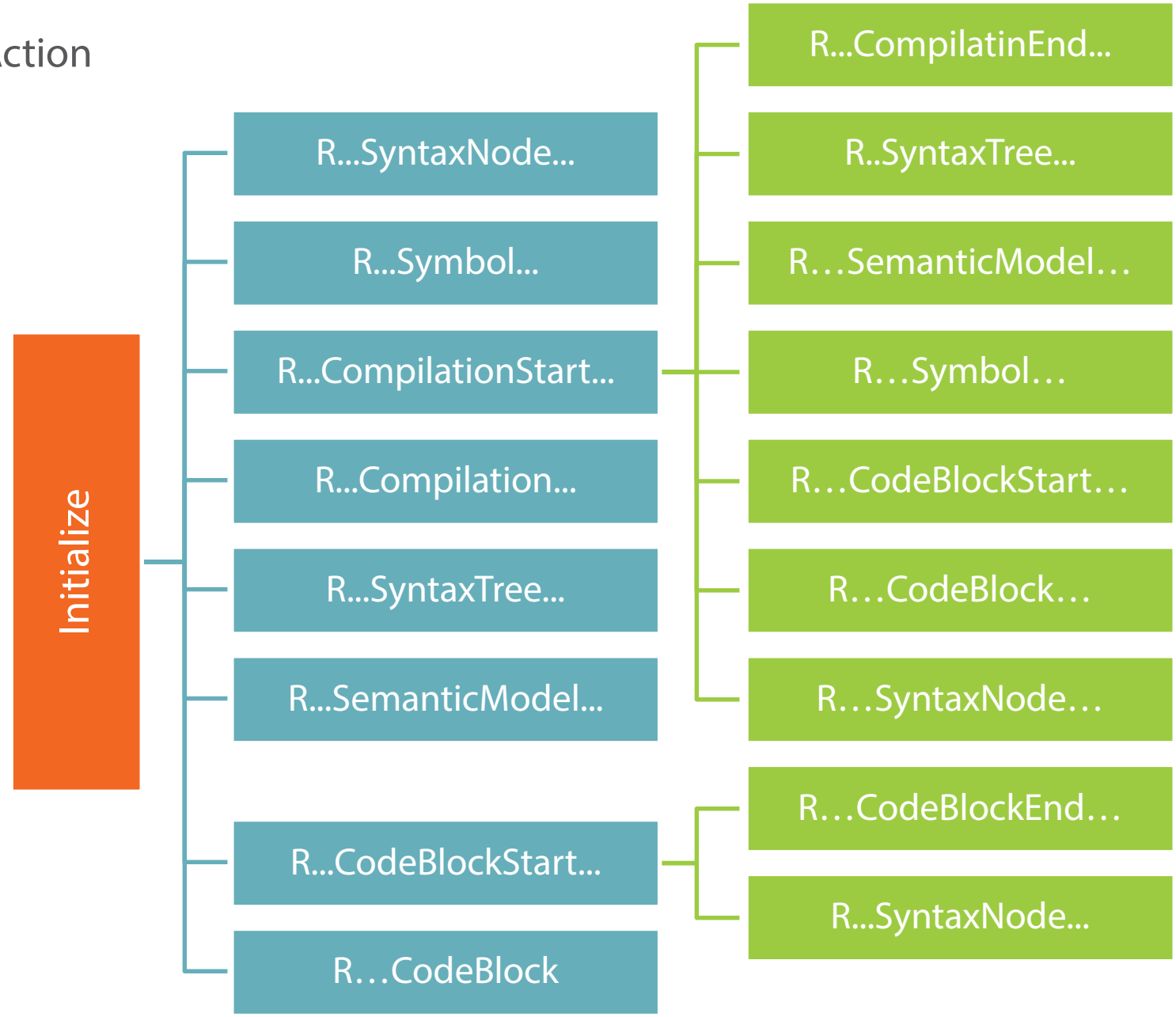
Register[ ]Action

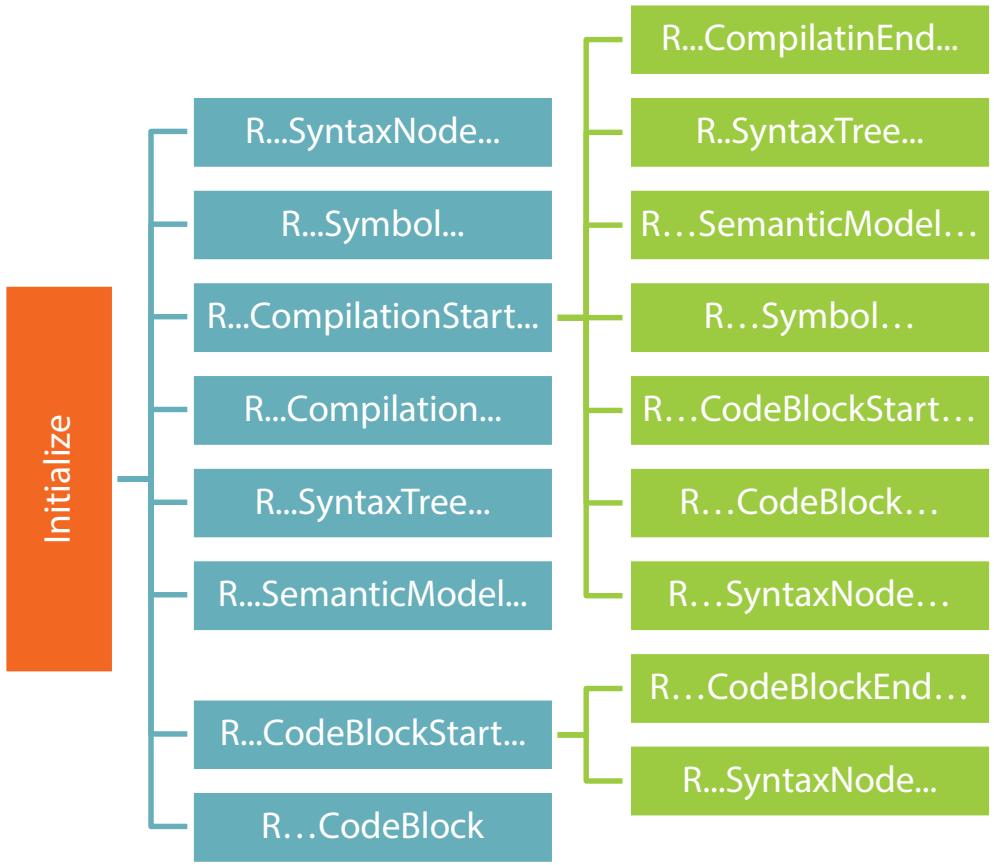


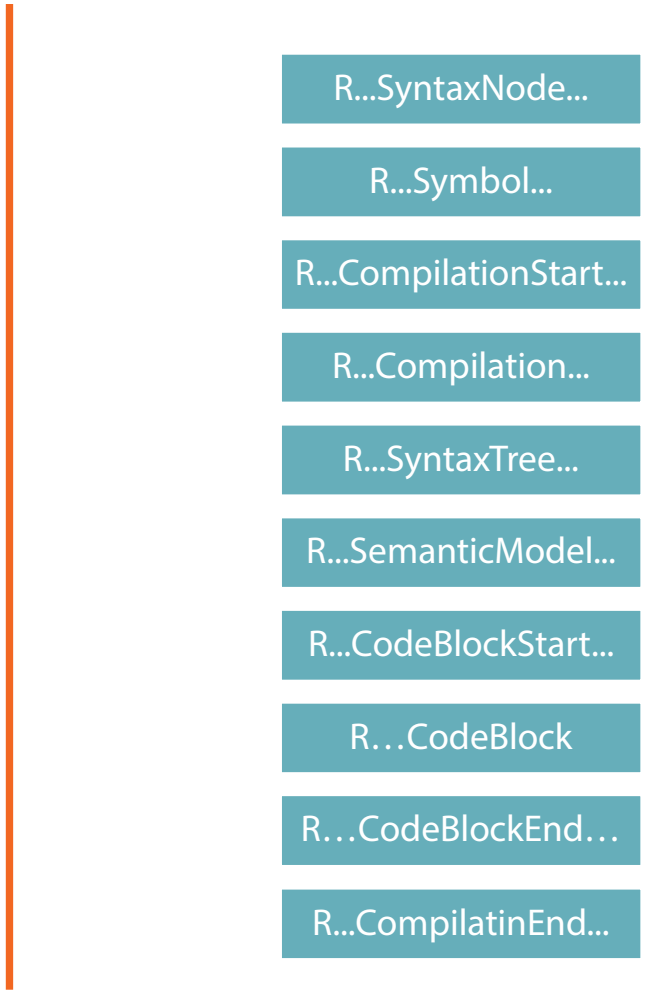
Register[ ]Action



Register[ ]Action







R...SyntaxNode...	(Action<SyntaxNodeAnalysisContext> action, ImmutableArray<TLanguageKindEnum> syntaxKinds)
R...Symbol...	(Action<SymbolAnalysisContext> action, ImmutableArray<SymbolKind> symbolKinds)
R...CompilationStart...	(Action<CompilationStartAnalysisContext> action)
R...Compilation...	(Action<CompilationAnalysisContext> action)
R...SyntaxTree...	(Action<SyntaxTreeAnalysisContext> action)
R...SemanticModel...	(Action<SemanticModelAnalysisContext> action)
R...CodeBlockStart...	(Action<CodeBlockStartAnalysisContext<TLanguageKindEnum>> action)
R…CodeBlock	(Action<CodeBlockAnalysisContext> action)
R…CodeBlockEnd…	(Action<CodeBlockAnalysisContext> action)
R...CompilatinEnd...	(Action<CompilationAnalysisContext> action)

R...SyntaxNode...

R...Symbol...

R...CompilationStart...

R...Compilation...

R...SyntaxTree...

R...SemanticModel...

R...CodeBlockStart...

R…CodeBlock

R…CodeBlockEnd…

R...CompilatinEnd...

SyntaxNodeAnalysisContext

R...SyntaxNode...

SymbolAnalysisContext

R...Symbol...

CompilationAnalysisContext

R...Compilation...

R...CompilatinEnd...

SyntaxTreeAnalysisContext

R...SyntaxTree...

SemanticModelAnalysisContext

R...SemanticModel...

CodeBlockAnalysisContext

R…CodeBlock

R…CodeBlockEnd…

Analysis Contexts that include ReportDiagnostic

```
public string Foo(int i)
{
    return i.ToString();
}
```

```
public string Foo(int i) => i.ToString();
```

```
public string Foo
{
    get
    {
        return "Hello World" + i;
    }
}
```

```
public string Foo => "Hello World" + i;
```

```
public void Foo()
{
    Bar();
}
```

```
public void Foo() => Bar();
```

> Code refactoring is the process of restructuring existing computer code – changing the *factoring* – without changing its external behavior.

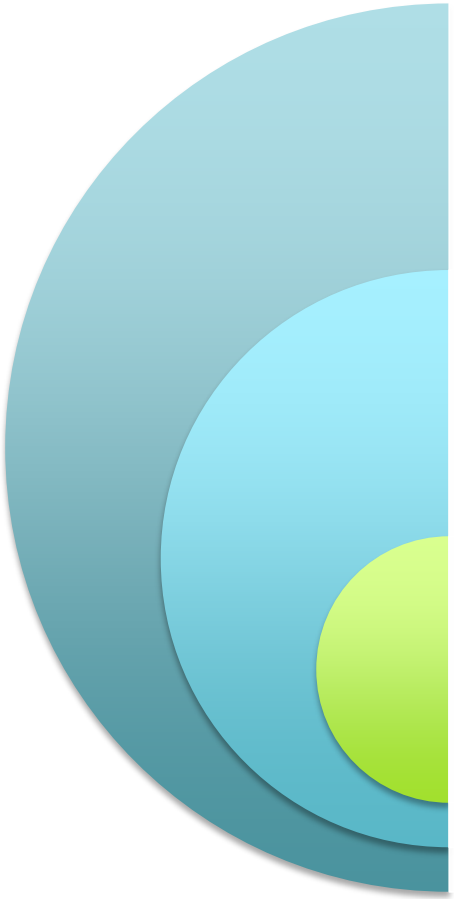
— http://en.wikipedia.org/wiki/Code_refactoring

…Its heart is a series of small behavior preserving transformations

— http://refactoring.com/

…The noun "refactoring" refers to one particular behaviour-preserving transformation, such as "Extract Method" or "Introduce Parameter".

— http://guide.agilealliance.org/guide/refactoring.html



Restructuring code to improve it while preserving original functionality, manually or automated

One of a defined series of change patterns that are often automated

A feature of .NET diagnostics

Refactoring

But wait!

Haven’t all our analyzers and code fixes been small automated changes that improve without changing what the code does?

	Analyzer/ Code Fix	Refactoring
Something is identifiable in code	*Required*	*Not required*
Clear or commonly accepted bad effect	*Use this*	
Squiggle should be allowed/configurable	*Use this*	*No squiggle*
Automated fix is available	*Not required*	*Required*
Lightbulb should appear	*Always*	*No lightbulb*
Analysis should occur during build	*Use this*	*Never*
User input is needed	*Bad*	*Use this*
Analysis is slow or expensive	*Bad*	*Use this*
Project specific	*Use this*	

Common Scenarios

Analyzer/Code Fix

- "This is a rule I want my entire team to see"
- "I want it to run as part of the build, in or out of Visual Studio"
- Deliver it per project with NuGet

Can be delivered easily via NuGet or VSIX

Refactoring

- "This is a convenience feature I would like available to me"
- "There's no analysis, it makes no sense outside Visual Studio"
- Deliver it machine wide with VSIX

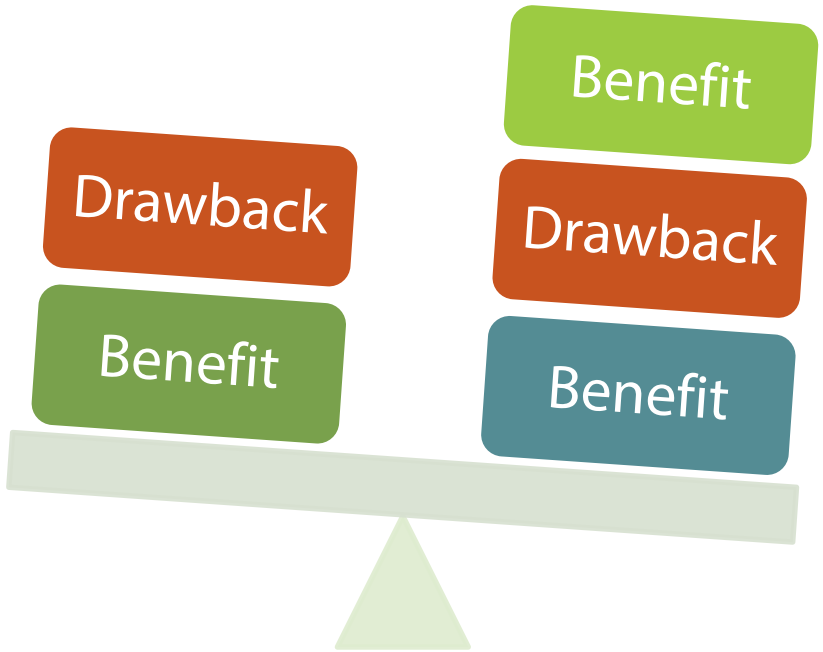
Can only be easily delivered via VSIX

Use an Analyzer/Code Fix unless you can't

	Analyzer/ Code Fix	Refactoring
fiable in code	*Required*	*Not required*
ccepted bad effect	*Use this*	
be allowed/configurable	*Use this*	*No squiggle*
Automated fix is available	*Not required*	*Required*
Lightbulb should appear	*Always*	*No lightbulb*
Analysis should occur during build	*Use this*	*Never*
User input is needed	*Bad*	*Use this*
Analysis is slow or expensive	*Bad*	*Use this*
Project specific	*Use this*	

		Analyzer/ Code Fix	Refactoring
Something is identifiable in code	Yes	*Required*	*Not required*
Clear or commonly accepted bad effect	No	*Use this*	
Squiggle should be allowed/configurable	No	*Use this*	*No squiggle*
Automated fix is available	Yes	*Not required*	*Required*
Lightbulb should appear	Yes	*Always* ✓	*No lightbulb*
Analysis should occur during build	No	*Use this*	*Never*
User input is needed	No	*Bad*	*Use this*
Analysis is slow or expensive	No	*Bad*	*Use this*
Project specific	Machine & project	*Use this*	

Refactorings



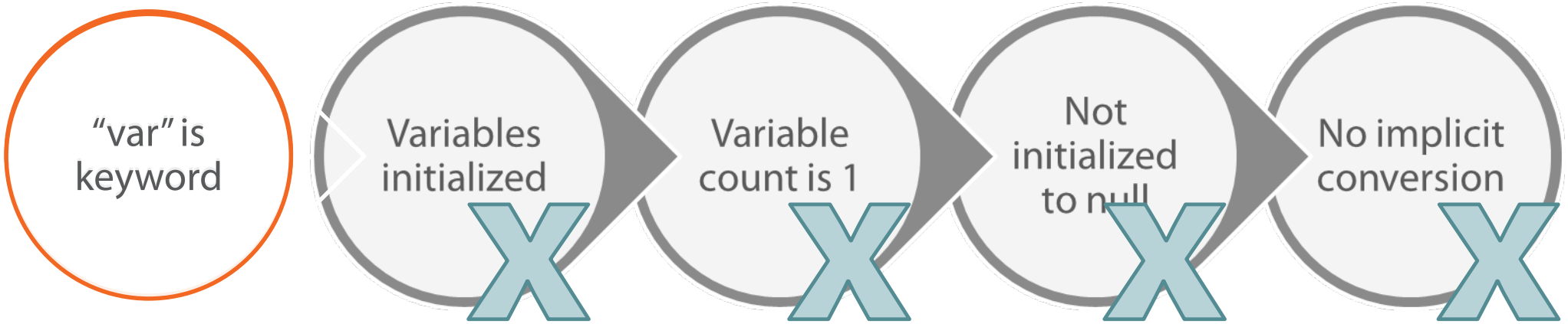
var int

Clarifying the Rules

Explicit Type to var

Clarifying the Rules

var to Explicit Type



Summary

Other Methods

Analyzers offer a number of registration methods

- Scope, scenario and state support
- Code blocks and compilation offer start/stop

Language Version Specific

Registering analyzers in compilation start

- Lets you respond to context, like language version
- Create analyzers for C# 6 features

Refactorings

Logical refactoring

- Analyzer: hidden severity
- Lightbulb

Roslyn Refactoring

- Different restrictions
- No lightbulb