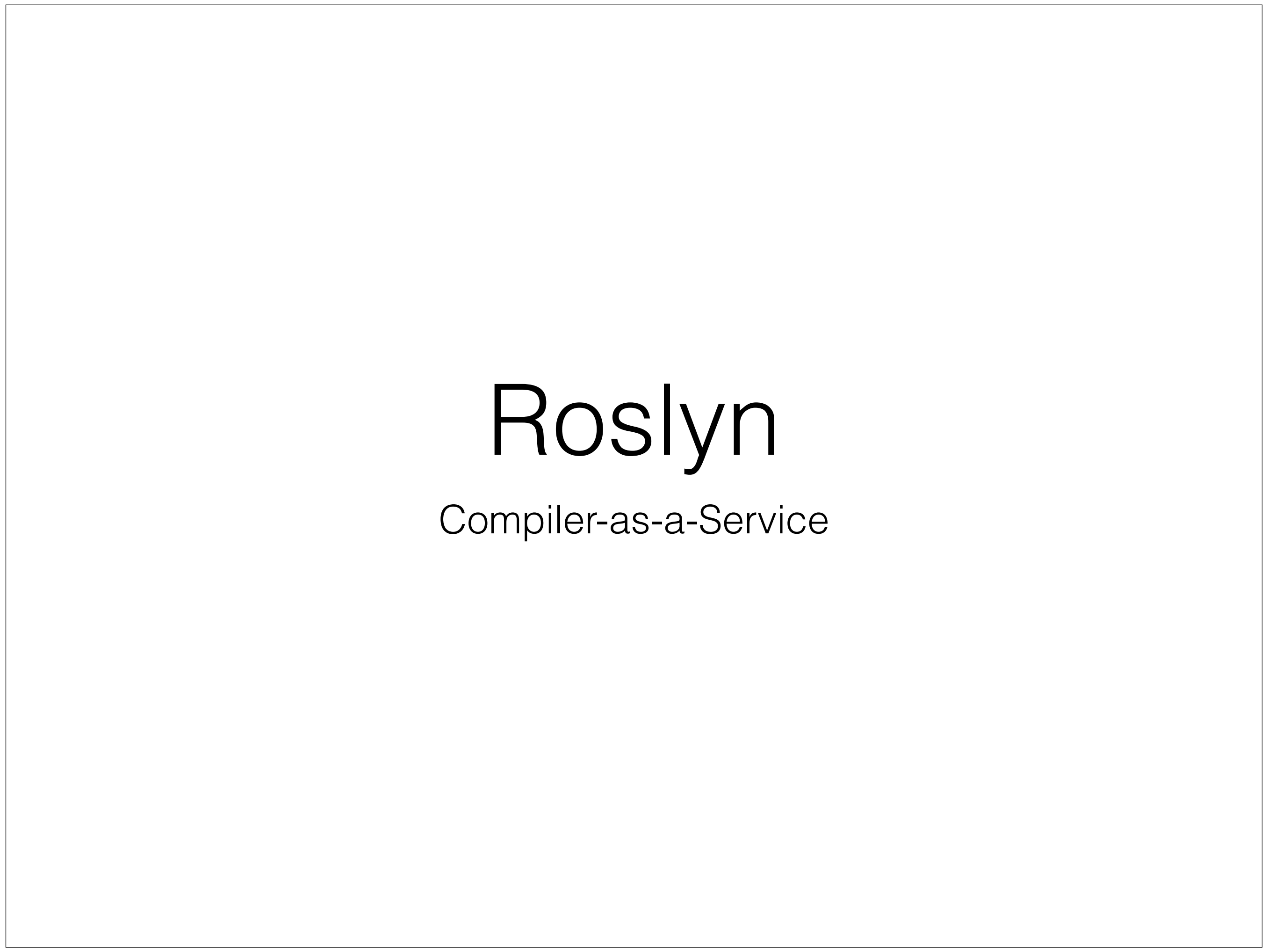


Roslyn

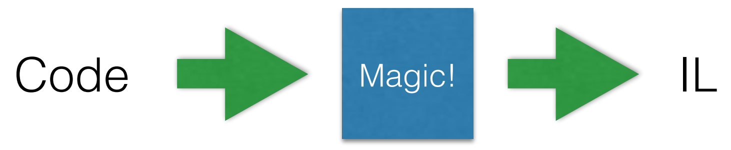
Compiler-as-a-Service

Today I want to talk about the C# compiler and how it fits into Visual Studio.

Touch briefly on how it currently is (and was) and what is going to be in VS14

The olden days

- Black-box compiler



- Getting info on code is hard

1) Visual Studio has been around since 2002

2) With the current platform its hard to get the semantics of code you want to refactor. ReSharper built their own parser and code model for this.

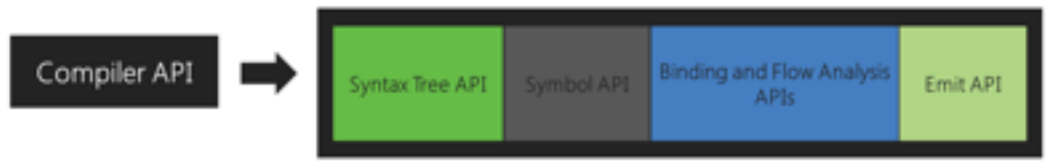
Out with the old,
In with the new

- C# 5.0 / 6.0
- IDE experience
  - IntelliSense
  - Code analysis
- Open Source

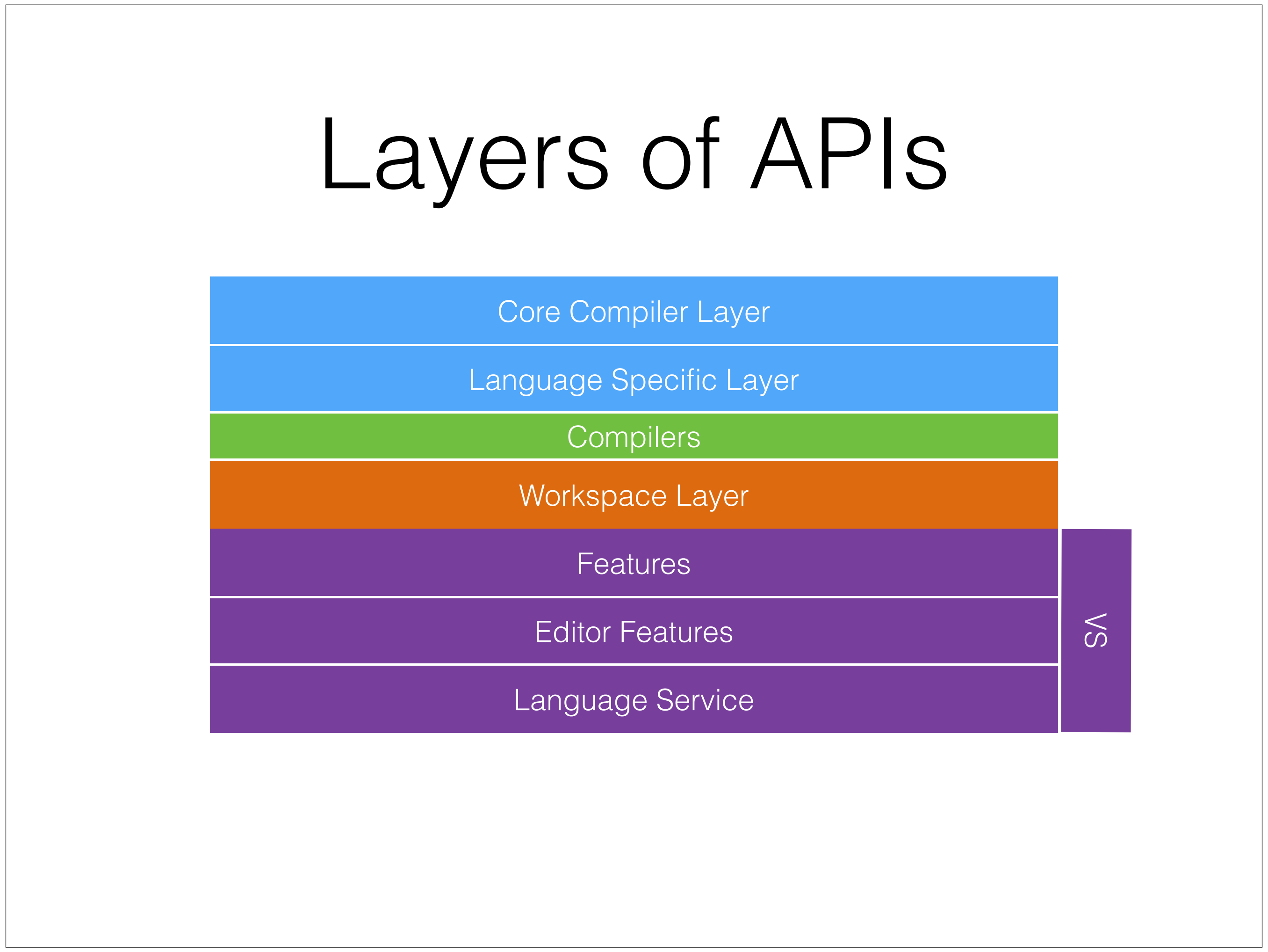
1) Features hard to implement (async in 5.0, newer features in 6.0, like exception filters)
2) Improvements to Visual Studio (14)
3) Microsoft is on the open source bandwagon as of late (ASP.Net etc) code is on Github
4) Work on Roslyn already started back in 2009

Compiler-as-a-Service

- Compiler internals as APIs

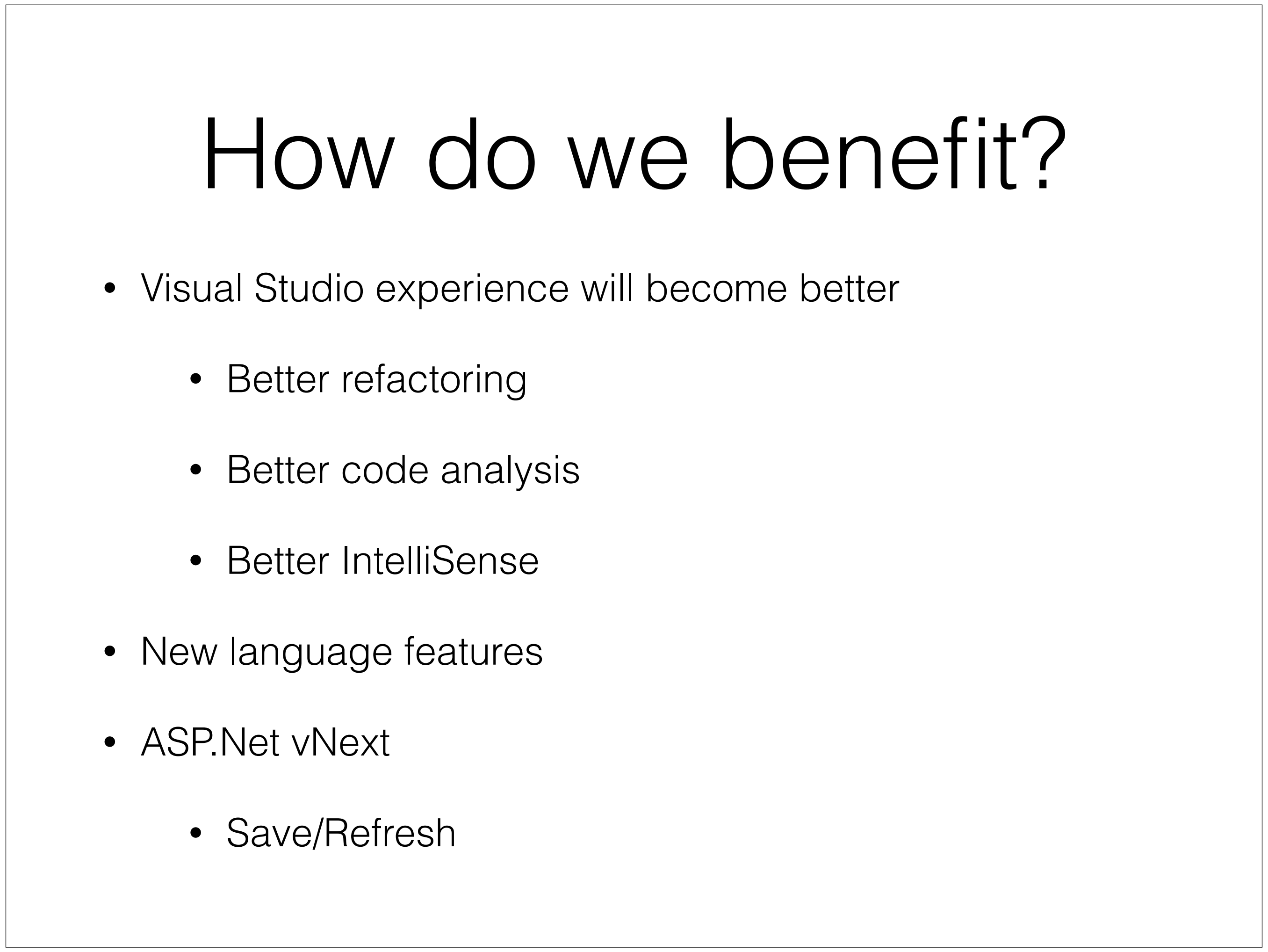


1) Each logical block (phase, color) has its own object model.
2) Also a semantic model per phase/block



Layers of APIs

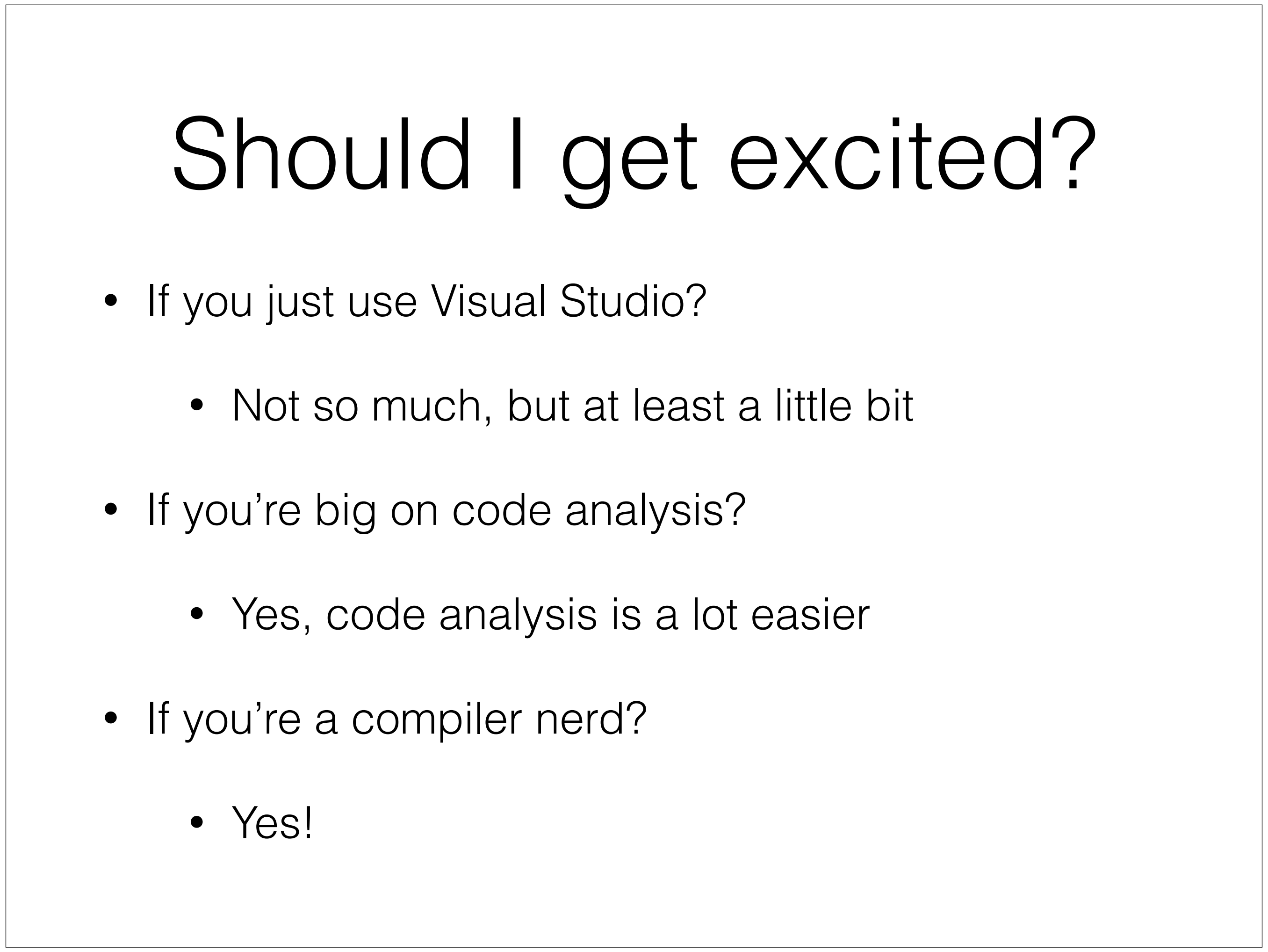
1) This defines the code and semantic models and all the types needed for syntax trees and such
2) This is what converts source code into the code model
3) The actual CLI and MSBuild task to utilize the previous two layers
4) The workspace layer represents your code (all of it), tracks changes and is the gateway from refactorings to actual code
5) This is all the glue that ties Roslyn to Visual Studio (refactoring, code analysis, GoTo definition, interop logic)



How do we benefit?

- Visual Studio experience will become better
    - Better refactoring
    - Better code analysis
    - Better IntelliSense
- New language features
- ASP.Net vNext
    - Save/Refresh

1) Yes but don’t we have ReSharper already? Yes, but remember what a resource hog that is.
2) Refactoring engine has been rebuilt
3) IntelliSense now actually knows what is valid code



Should I get excited?

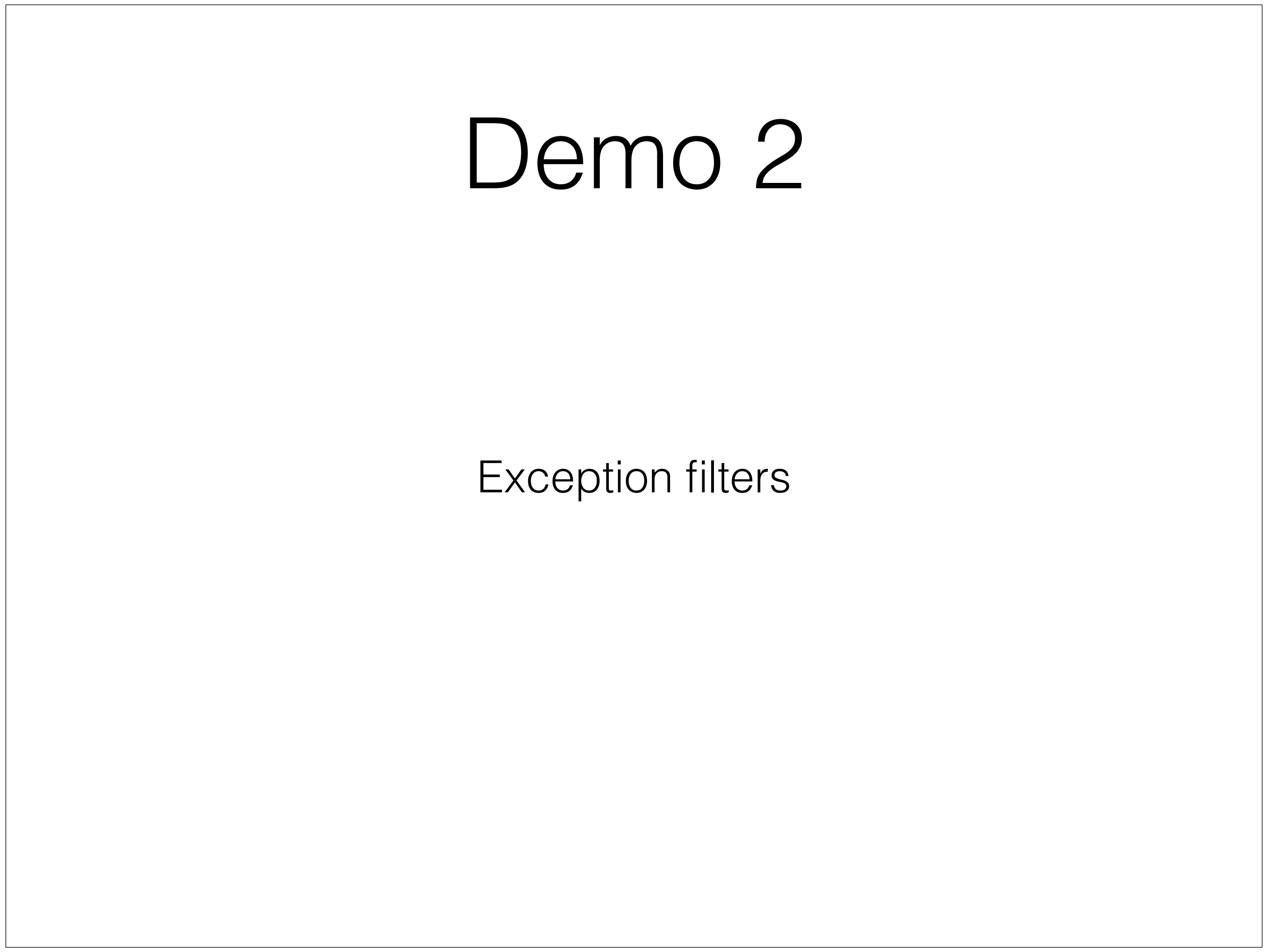
- If you just use Visual Studio?
  - Not so much, but at least a little bit
- If you're big on code analysis?
  - Yes, code analysis is a lot easier
- If you're a compiler nerd?
  - Yes!

1) More refactorings available
2) Create and deploy rules more easily
3) If you want to do AOP (you can rewrite source code with this)

Demo 1

Code analysis

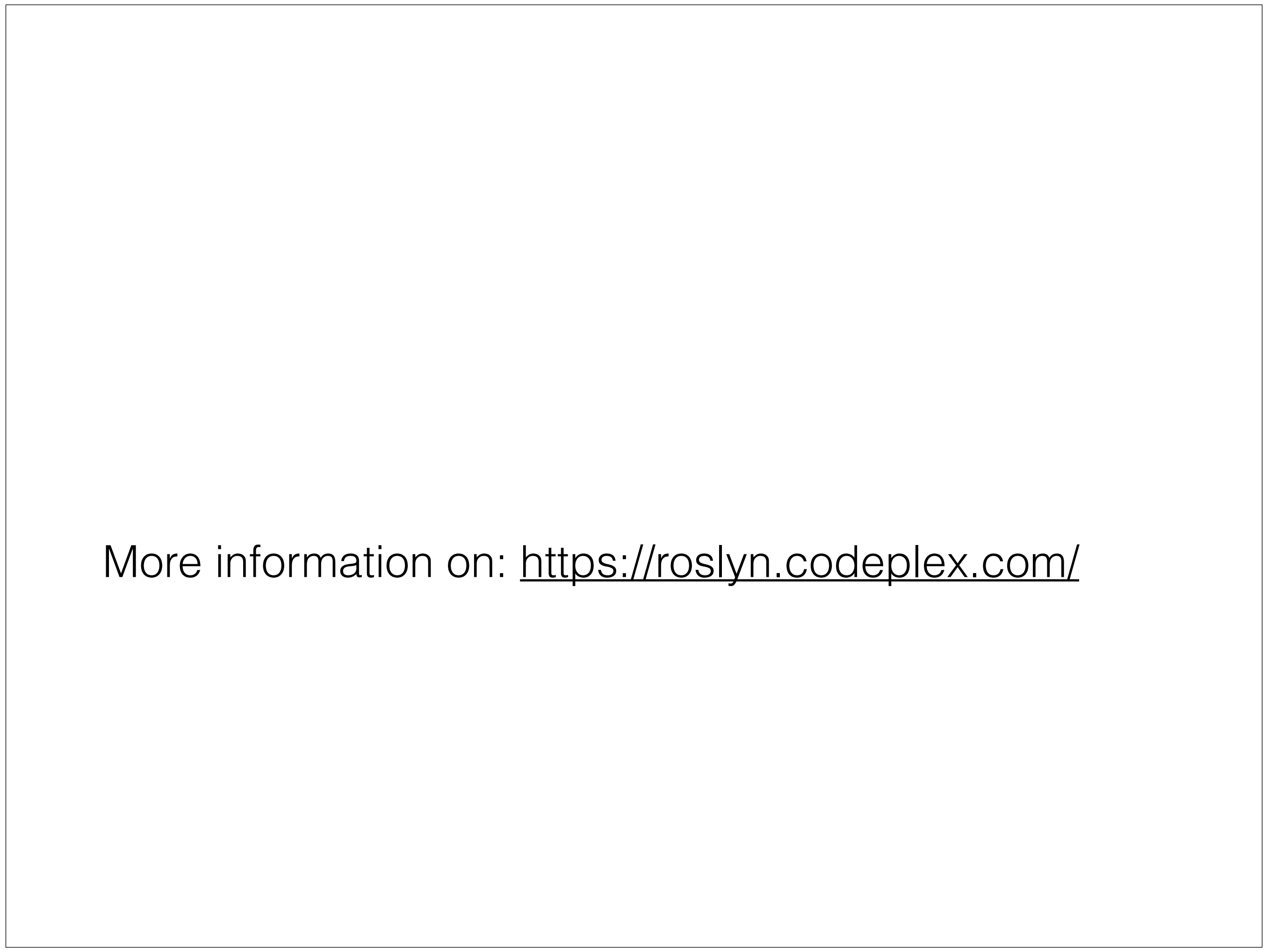
1) Show SonarCompanion solution, Messages
2) Create a new analyzer for types that don’t end in Message
3) Create a code fix
4) Utilize rule in SonarCompanion solution through NuGet



Demo 2

Exception filters

1) Preserves callstack
2) Especially useful for crashes on a live system where you get a crashdump file. Previously the callstack was incomplete giving you no info at all
3) Show off refactorings


More information on: https://roslyn.codeplex.com/

- Mono will also use the new compilers (Roslyn is open source remember)