Roslyn FAQ

This document contains FAQs and good learning or getting-started questions. It is up to date for Roslyn September CTP3.

Contents

Introduction

This FAQ is inspired by all the great questions and answers from previous Roslyn CTPs. In several cases, customers were helping each other without the Roslyn team chiming in, so great job everyone! This FAQ is a collection of great learning questions (little how-to recipes) and questions actually frequently asked over the lifetime of the project. You will find many good pointers by searching for keywords or phrases in this document.

Many answers have code snippets associated with them. The code exists as samples/tests that we release with the CTP – "%userprofile%\Documents\Microsoft Roslyn CTP - September 2012\{CSharp|VisualBasic}\APISampleUnitTests{CS|VB}\faq.{cs|vb}". Where there is code available, the answer to the question has one or more tags such as "FAQ(27)" in the text. You can open the file named above and/or the associated project and search the file for the tag. By not listing all the code in this document, the document is less likely to diverge from working code. The samples/test project is always up to date with any API changes.

Project / Cross-cutting Questions

What docs are available on Roslyn?

First, please see the MSDN release site, http://msdn.com/roslyn, for a number of links to walkthroughs, overview docs, and blogs. There are specs for the VS REPL, Roslyn Scripting APIs, and some compiler APIs at the bottom of this page.

Does installing Roslyn work SxS with regular Visual Studio 2010 SP1 or Visual Studio 2012?

Roslyn CTP3 only targets Visual Studio 2012 RTM, and it does not install for VS 2010 (including SP1).

Will Roslyn be open source beyond examples and walkthroughs?

There are no firm plans regarding open source and the Roslyn compilers or APIs, but we are considering how to provide the best resources.

Can I rewrite source code within the compiler pipeline?

Roslyn does not provide a plug-in architecture throughout the compiler pipeline so that at each stage you can affect syntax parsed, semantic analysis, optimization algorithms, code emission, etc. However, you can use a pre-build rule to analyze and generate different code that MSBuild then feeds to csc.exe or vbc.exe. You can use Roslyn to parse code and semantically analyze it, and then rewrite the trees, change references, etc. Then compile the result as a new compilation.

Can I redistribute the Roslyn DLLs with my samples or code on my blog?

For sample code or learning purposes, the recommended way to redistribute the Roslyn DLLs is with the Roslyn Nuget package, http://www.nuget.org/packages/Roslyn . As the Roslyn CTP is in the very early stages, it's intended for evaluation and gathering feedback, not for any actual product development.

Does the Roslyn CTP come with a "Go Live" EULA?

No. Due to the early stages of development, the CTP is for gathering feedback and community involvement in meeting requirements. The CTP is not ready for betting on real product development.

Where can I see the Roslyn EULA after installation or without committing to install Roslyn?

The license is available online. You can find it on the NuGet Roslyn package release page at: http://www.nuget.org/packages/Roslyn and here.

How do the Roslyn APIs relate to the VS Code Model and CodeDom?

The CodeDom targets programmatic code generation and compilation scenarios in ASP.NET on the server. It was later co-opted for some tooling uses, adding modeling of existing code to the code generation functionality.

The VS Code Model tries to relieve ISVs from having to parse code in VS so that ISVs can provide code-oriented extensions to VS. The VS Code Model represented code (and had meager code generation support) down to types, members, and parameters. It did NOT go into functions to the statement level.

The Roslyn APIs fully model all C# and VB code and provide full code generation or updating capabilities. Ultimately, we will re-implement the VS Code Model on the Roslyn APIs. The CodeDom is a separate technology built outside the C# and VB teams, and it does not need to be rewritten (though someone may want to move it to the new Roslyn APIs).

Can you just open a Connect bug for me?

As Microsoft employees chatting with folks on forums or via email, we're happy to open internal bugs to save you effort. If we say we'll open a bug, then you're guaranteed we opened and gave the issue every bit of weight that we would give any customer issue. If you want to track the bug and be automatically notified of any changes to the bug, then we need you to open a Connect bug.

There are a couple of key reasons we ask you to open Connect bugs, rather than our doing it for you. First, we try to avoid artificially driving up customer demand on our own features, so sometimes when reviewing Connect bugs, auditors ask us when a bug doesn't appear to come from customers. Second, the Connect bug keeps a line open to the customer reporting the issue in case we have follow up questions about repro info or scenario clarifications, etc.

When an employee logs the Connect issue for you, any updates cause Connect to send mail to the employee, not the customer, which is a problem should the bug go to another team from the employee who opened the bug. The customer info or contact may be lost. That said, this isn't a hardcore policy, and there are targeted cases where we will open a Connect bug on behalf of a customer when we have a long-term partner relationship with them. This isn't our general workflow.

We do want to make it super easy for customers who have given us feedback to log that feedback though, so we tend to offer to log the issue for you. However, if you want to get update mail and track the bug, we ask you to log the Connect bug. There is a VS feedback tool that is designed to make it even easier to report Connect issues (http://visualstudiogallery.msdn.microsoft.com/f8a5aac8-0418-4f88-9d34-bdbe2c4cfe72).

Getting Information Questions

How do I get type info for a variable in a declaration, with inferred ('var') or explicit variable type?

See the sample code answer tagged "FAQ(1)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see how to get the type from the IdentifierNameSyntax modeling 'var' as an identifier (C#) or the type name as an identifier (VB). See the sample code answer tagged "FAQ(2)" to see how to get the type from the VariableDeclaratorSyntax modeling an entire 'var' declaration (C#) or a 'dim' declaration with Option Infer on (VB).

How do I get all variables declared of a specified type that are available at a given code locations?

See the sample code answer tagged "FAQ(4)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see a few examples of how to how to get the names of a specified type, including how to filter for different kinds of members or scopes. The SemanticModel.LookupSymbols API will give you all the Symbols that can be referenced by a simple name at a given location. There are a few issues to be aware of.

LookupSymbols may return symbols with unutterable names in C#, such as for anonymous types. You can check the Symbol property CanBeReferencedByName.

The results include both local symbols and wider scopes such as symbols for methods, fields, types, and namespaces. If you only care about local variables, you can check the Kind of the returned Symbols, see the code example. Also, SemanticModel.LookupSymbols has overloads that allow you to specify LookupOptions, for example, whether to include extension methods.

LookupSymbols may return symbols that cannot be referenced, but they are technically in scope. For example, since C# lifts lexicals to function scope, so you could find the symbol for a local declared later in the function but that is illegal semantically to reference before the declaration.

How do I get a completion list or accessible symbols at a code location?

See the sample code answer tagged “FAQ(4)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see a few examples of how to get accessible names at a location, including how to filter for different kinds of members or scopes. This question is similar to “How do I get all variables declared of a specified type that are available at a given code locations?”, but note in that questions answer that truly getting what Visual Studio shows in a completion list has multiple issues that you need to deal with.

How do I get a completion list with members of an accessible type?

See the sample code answer tagged “FAQ(5)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see how to get accessible members.

How do I get caller/callee info?

See the sample code answer tagged “FAQ(6)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see how to get caller/callee information. Note, the sample is not necessarily complete; for example, the analyzed code could have assigned the function to a delegate variable and then invoked it, for which the sample does not account.

How do I go from an ISolution to Find All References on a symbol/type?

See the sample code answer tagged “FAQ(7)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see how to get references. Note, to use convenience extension methods, such as FindReferences, you need to use the Roslyn.Services namespace and reference Roslyn.Services.dll and Roslyn.Services.{CSharp|VisualBasic}.dll.

How do I find all calls in a compilation into a particular namespace?

See the sample code answer tagged “FAQ(8)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see how to find all calls into a particular namespace (or to functions from that namespace).

How do I get all symbols of an assembly (or all referenced assemblies)?

See the sample code answers tagged “FAQ(9)” and “FAQ(10)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb). The former shows walking all namespace symbols, then type symbols, and then for simplicity walking field and method definitions. The latter shows a walker for all expressions to discovered referenced symbols.

How do I get the type of an expression node?

See the sample code answer tagged “FAQ(3)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see a few examples of getting an expression’s type information.

How do I get type information of parameters and declared locals with common API?

See the sample code answer tagged “FAQ(3)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see a few examples of getting variable type information. You’ll note Roslyn models parameters and other locals differently, with LocalSymbol and ParameterSymbol, which makes sense since parameters can be ref/out, have default values, etc.

How do I get the type information (TypeSymbol) from a semantic model for an identifier (or an IdentifierNameSyntax node)?

See the sample code answer tagged “FAQ(1)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see how to get the type from the IdentifierNameSyntax modeling ‘var’ as an identifier.

How do I compare syntax nodes (optionally ignoring attached trivia)?

See the sample code answer tagged “FAQ(11)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see several aspects of comparing trees and nodes.

How are comments stored in the syntax tree (and how to use the Syntax Visualizer)?

Roslyn trees store comments as trivia on tokens. The heuristic used is to attach trivia to the token following it (that is, as leading trivia on the node containing the token). However, if the trivia is after the last token on a line, it attaches to the preceding token (that is, as trailing trivia).

In the following example, the first comment attaches as LeadingTrivia to the 'int' token in the declaration following it. You can also see the leading trivia on the LocalDeclarationStatementSyntax node containing the 'int' token. The first comment trivia attaches to the first token following it even though 'int' is on a different line because there are no tokens on the line with the comment.

```
int method1 () {
    int a = 1;
    // comment1
    int b = 1; // comment 2
 }
```

The second comment attaches as TrailingTrivia to the semi-colon, and you can also see it on the LocalDeclarationStatementSyntax node.

For a good explanation of Visualizers to inspect pieces of code and how Roslyn constructs syntax trees, go here. The Roslyn CTP ships a visualizer you can use in Visual Studio, which lets you browse a tree control showing the syntax nodes, tokens, and trivia attached to them.

See the sample code answer tagged "FAQ(29)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see a little SyntaxWalker that collects the comments in code and distinguishes structured doc comments.

What is structured trivia and how do I get at it?

Most trivia (such as WhitespaceTrivia or SingleLineCommentTrivia) simply distinguishes the kind of trivia with the Kind property. All trivia have the same struct type, SyntaxTrivia.

Some trivia (such as directives or xml doc comments) have structured models with SyntaxNode derived types to represent them. You can check whether a trivia has structure by checking the HasStructure property. If a trivia has structure, you can call the GetStructure method on the trivia to get a SyntaxNode. For example, if you have an IfDirective trivia, calling its GetStructure method returns a IfDirectiveSyntax node. This node gives you access to the HashToken, IfKeyword, condition's ExpressionSyntax node, etc.

Is there a syntax tree visualization or tools to visually inspect a tree?

There are the Syntax Visualizer Tool Window and the Syntax Debugger Visualizer. The Syntax Visualizer Tool Window sample is a Visual Studio Extension that enables inspection of 'live' Roslyn Syntax Trees for any C# or VB code file that is open inside the Visual Studio IDE when the Roslyn Language Service is present. The Syntax Debugger Visualizer sample is a tool extension to VS that you can use within a Visual Studio debug session to visualize Roslyn Syntax Trees, Syntax Nodes, Syntax Tokens, and Syntax Trivia.

See how to use the visualizer tool window and the pop-up debug visualizer from the autos/locals window:

http://blogs.msdn.com/b/visualstudio/archive/2011/10/19/roslyn-syntax-visualizers.aspx

How do I tell if the type associated with a symbol is a known type? Do I have to construct the AssemblyQualifiedName myself?

It is best to use Equals(). Symbols overload operator==, which works in many cases, but doesn't work properly with the Common Symbol API (ISymbol). The easiest way to compare a known type to a symbol's type is to use Compilation.GetTypeByMetadataName to get the known type, and then compare with Equals(). See the sample code answer tagged "FAQ(12)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see symbol comparison.

For a small, fixed set of very common types (for example, like Int32 and String) and a few special types of interest to the compiler, the property TypeSymbol.SpecialType lets you easily see if your known type is one of those special types. See the sample code answers tagged "FAQ(1)" and "FAQ(3)" to see symbol comparisons to special types.

Comparing symbols from different compilations has undefined results.

How do I tell if Symbols are the same?

It is best to use Equals(). Symbols overload operator==, which works in many cases, but doesn't work properly with the Common Symbol API (ISymbol). See the sample code answer tagged "FAQ(12)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see symbol comparison.

There are no guarantees that APIs return the same Symbol object even when called on the exact same argument objects.

For a small, fixed set of very common types (for example, like Int32 and String) and a few special types of interest to the compiler, the property TypeSymbol.SpecialType lets you easily see if a type symbol is one of those special types. See the sample code answers tagged "FAQ(1)" and "FAQ(3)" to see symbol comparisons to special types.

Comparing symbols from different compilations has undefined results.

How can I use the same icons that C# and VB use in completion lists?

You can MEF-import a Microsoft.VisualStudio.Language.IntelliSense.IGlyphService interface. The interface documentation is here. That's the same Visual Studio interface that we use to retrieve glyphs to display in IntelliSense.

To MEF-import, it looks like this:

```
[ExportCompletionProviderAttribute("Completion", LanguageNames.CSharp)]
    public class CompletionProvider :  ICompletionProvider {
        [Import]
        public IGlyphService m_glyphService;
```

When MEF activates the CompletionProvider, it sets the service property to an instance of the service.

How can I test if a semantic model can provide information about a syntax node?

See the sample code answer tagged “FAQ(13)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see several ways of checking if trees, nodes, and semantic models are related.

How can I get the metadata token for an ISymbol?

This might be interesting for interoperating between Roslyn APIs and other tools such as ildasm. We do not have a way to do this today.

What's with identifier's (SyntaxTokens) having Value rather than Name, and how do Value, ValueText, and GetText relate?

When I consider "ParameterList.Parameters[0].Identifier.Value", wouldn't Name be a better choice than Value?

Regarding .Value vs. .Name for identifiers, we use SyntaxToken to represent identifiers. SyntaxToken also represents all significant pieces of text in source code, such as literals, punctuation, etc. SyntaxTokens have no concept of a name, but do model text. SyntaxToken.GetText() returns this text exactly as it was read when parsing the source code. SyntaxToken.Value returns the object "value" of the text represented by the SyntaxToken. SyntaxToken.ValueText returns the text representation of this value.

Consider "long @long = 1L". SyntaxToken.Value and SyntaxToken.ValueText for the escaped identifier "@long" both return the string "long". However, SyntaxToken.GetText() returns the string "@long". Similarly, SyntaxToken.Value for the literal "1L" returns the boxed integer "1". SyntaxToken.ValueText returns the string "1". SyntaxToken.GetText() returns the text parsed from the code, "1L".

See the sample code answer tagged "FAQ(14)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb).

Why use ChildNodesAndTokens rather than just Children?

There was a Children property originally. The feedback from usability testing showed that a key problem people had using the syntax API was using this collection. The names were changed to emphasize the distinction between nodes and tokens and to steer people away from using the SyntaxNodeOrToken type by default.

How do I get line and column information to report errors?

See the sample code answers tagged "FAQ(16)" and "FAQ(17)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see several examples of fetching position information from trees and nodes.

How do I find all syntactic sub trees of a particular kind?

See the sample code answer tagged "FAQ(18)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see a SyntaxWalker that visits specific kinds of nodes and tokens. Several other samples use node.DescendentNodes().OfType<>() to fetch sub trees (for example, "FAQ(1)"). Some samples show filtering with node.DescendentNodes().First(t => t.Kind == SyntaxKind…), such as "FAQ(11)".

How do I get the fully qualified type name of a definition?

See the sample code answer tagged “FAQ(19)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see several examples of getting and comparing names, including open generic types vs. closed generic types.

How do I determine which overload binds at a given call site?

See the sample code answer tagged “FAQ(20)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see resolving overloads.

How can I tell if a TypeSymbol from an expression can be assigned to the TypeSymbol of a location?

See the sample code answers tagged “FAQ(21)” and “FAQ(22)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see several examples of determining what’s assignable and if so what sort of conversion would be needed if any.

How can I get a fully qualified type name for a local variable declaration, instead of just the text parsed in the SyntaxTree?

See the sample code answer tagged “FAQ(19)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb).

How do I get the .NET Framework version?

See the sample code answer tagged “FAQ(23)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb), but essentially you get the containing assembly of SpecialType.System_Object and get the assembly’s version.

How do I get a project's assembly symbol, references, and syntax trees for each document or item in the project?

See the sample code answer tagged “FAQ(24)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see how to do it for one reference and document, but you can easily convert the code to loop over all.

How do I extract an annotation of a particular (sub) type?

See the sample code answer tagged "FAQ(25)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see annotating certain tokens and then finding tokens with a particular type of annotation. The sample doesn't show it, but this works with nodes as well.

Why doesn't the Workspace API support more VS concepts (nested documents or non-code files)?

The Workspace API represents the view of solutions, projects, and documents that the C# and VB language services actually consume. The Workspace model includes C# and VB source files, references, compilation options, and other similar details. However, the Workspace API does not represent every feature of the Visual Studio project system, such as modeling xaml, resource files, etc. The Workspace model also works independently of Visual Studio.

To get all the project item information that Visual Studio has, use the MSBuild APIs which ship in the .NET Framework. If your code executes within Visual Studio, you could also use the IVsHierarchy API to enumerate project items, which would expose access to nested documents that Visual Studio manages. If you want to do semantic analysis of C# and VB files, you should use a workspace.

How do I get base type or implemented interface information and which members override or implement base members?

See the sample code answers tagged "FAQ(37)" "FAQ(38)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see fetching types by name, fetching base and implemented types, and inspecting members for information such as virtual, override, sealed, etc.

Constructing and Updating Tree Questions

How do I add a method to a class?

See the sample code answer tagged "FAQ(26)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb).

How do I replace a sub expression, declaration, etc.?

See the sample code answers tagged "FAQ(26)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) for replacing a class declaration and "FAQ(27)" for replacing sub expressions.

How do I get an IWorkspace from my RefactoringProvider?

Please see the walkthroughs for creating features in Visual Studio. Once you have an extension that's loaded, you can access workspaces. For example, if you have a RefactoringProvider, when Visual Studio calls on it, VS invokes ICodeRefactoringProvider.GetRefactoring() or ICodeIssueProvider.GetIssues(), passing an ITextBuffer. You can get the workspace with Workspace.GetWorkspace(myITextBuffer.AsTextContainer()), and then you have a snapshot of the ISolution, IProject, and IDocument models that an IWorkspace manages. You will need usings for Roslyn.Services and Roslyn.Services.Editor.

Note, there may be multiple workspaces active in Visual Studio at any time if the Roslyn language service is loaded:

- If the Roslyn language service is loaded, there is a workspace modeling the active solution inside Visual Studio.
- If the Roslyn language service is loaded, there is a workspace modeling a solution and projects for any miscellaneous (see Misc Files Project) .cs and .vb files open in VS.
- There is a workspace modeling solution and projects for any miscellaneous .csx files open in VS.
- There is a workspace which models the submission chain in the Interactive window.

Because the Roslyn language service features (completion, code Issues, refactoring support, etc.) operate on an IWorkspace, the features work in all of the above workspace contexts.

How do I change the name of a symbol at the declaration site and all reference sites?

See the sample code answer tagged "FAQ(28)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see a renaming example. Note, this sample doesn't handle generic names, and depending on the type of thing you are renaming, you might create the SyntaxRewriter differently (for example, not visiting class declarations or constructor declarations).

Can I add custom information to syntax and symbols?

See the sample code answer tagged “FAQ(25)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see annotating certain nodes and then finding nodes with a particular kind of annotation.

There is no mechanism for Symbols.

How can I remove a statement with a SyntaxRewriter?

See the sample code answer tagged “FAQ(30)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see removing a statement and when you can simply return null from a visit method or when you need to return Syntax.EmptyStatement().

How do I construct a pointer type or array type given another type?

See the sample code answer tagged “FAQ(31)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb).

How can I remove #region and #endregion (structured trivia) with SyntaxRewriter?

See the sample code answers tagged “FAQ(32)” and “FAQ(33)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see a few ways of removing these directives.

How can I add logging to all statements of a particular kind (for example, to log contents of variables)?

See the sample code answer tagged “FAQ(34)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb).

How can I remove all comments from a file of code?

See the sample code answers tagged “FAQ(32)” and “FAQ(33)” (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see a few ways of removing #region directives, but you can change the code to look for SyntaxKind SingleLineComment or MutliLineComment.

Scripting, REPL, Executing Code, DLR, Iron* Questions

How does Roslyn relate to DLR Scripting?

Will Roslyn support the DLR Scripting Hosting API? What guidance is there which to use?

The DLR has a vetted hosting/scripting model (see the spec here for an overview or details). The current thinking is that the Roslyn Scripting APIs will be a minimal subset of the DLR Scripting Hosting APIs to support code execution. We want to enable .NET applications to use C# and VB as "scripting" languages for .NET apps. We might provide a working sample that uses the Roslyn Scripting APIs and other Roslyn APIs to support the DLR Scripting Hosting API model. If you wanted a .NET application that could be scripted by C#, VB, IronPython, IronRuby, or any DLR language, then you could take the sample to do that. Of course, by the time Roslyn is done, the Roslyn Scripting APIs could take on any form and even perhaps solve some problems or scenarios that the DLR Scripting Hosting APIs do not solve. For example, the DLR did not support some things such as host controls of stack depth or host controls of which APIs script code could access.

How do the Roslyn APIs relate to LINQ Expression Trees or Expression Trees v2? Is one better for meta-programming or implementing DSLs?

Expression Trees v2 are a semantic model with some shapes resembling syntax. They do support some control flow, assignment statements, recursion, etc. However, they do not model many things that the C# and VB languages have, for example, type definitions. The Roslyn APIs will have full fidelity with C# and VB for syntax, semantic binding, code emission, etc.

While the DLR and ETs v2 may lend themselves to some meta-programming, their real focus was supporting dynamic languages on .NET and enabling applications to host those languages for application scripting purposes.

The Roslyn APIs will be much better suited and probably your only consideration if your goal is embedded Domain Specific Languages (DSLs). Embedded DSLs are syntax extensions to a language for which you can supply core language code during compilation or prior to execution to give semantics to the new syntax. See the Dylan language for an example of DSL support in a language (http://opendylan.org/books/dpg/db_329.html). Otherwise, a DSL is just a formal language that is highly specific to a domain, such as SQL, as opposed to a general purpose language like C#. The Roslyn APIs will certainly be less limiting since they will support everything languages like C# and VB can express.

How do I get started with the Roslyn Scripting APIs?

After looking at the scripting introduction walkthrough and the Paint-like scripting walkthrough, you could check out Anoop Madhusudanan's fun console application that loads C# code as script code:

http://www.amazedsaint.com/2011/10/c-vnext-roslynan-introduction-and-quick.html

http://www.codeproject.com/KB/cs/csharpforscripting.aspx

How do I compile some code into a collectible type or DynamicMethod?

You can ultimately use something like Compilation.Emit(ModuleBuilder) to emit to a collectible assembly/type, but the most simple way will be to use ScriptEngine.CompileToMethod() which takes source as a string and returns a delegate. Unfortunately, Roslyn does not implement CompileToMethod() yet.

Can I hit F5 and then attach the REPL to a running process?

Unfortunately, we don't have a way to get the debugger context right now. The REPL also supports much richer exploratory and re-definition capabilities than EnC or .NET process debugging supports. We are aware that attaching the REPL to a .NET application would be cool and useful, and we will consider what can be done after core deliverables are solid.

How do I add references for my .csx files or to the REPL's execution context in VS?

You can use the #r directive to add references to assemblies. It takes a string argument that can be a simple assembly name ("System.Core"), full or partial path name, or fully qualified assembly name. See the REPL spec document here.

Csi.exe has an associated csi.rsp file in which there are default /r's for references.

There will eventually also be a default VS .rsp file you can designate and project-specific .rsp file contents you can describe in a project property page.

How do I add references for the code I'm executing with ScriptEngine?

You can execute snippets of the form "#r …", but perhaps a more natural way is to add the references when instantiating the Engine or adding them with AddReference on the engine or session. There are a few ways to do this from simple string names of assemblies (or pathnames to them) to creating

MetadataReferences. The references added to an engine apply to all snippet executions while the references added to a session only apply when that session is passed to an Engine.Execute method.

Note, the Roslyn Scripting APIs are a first cut and still need more design work to flesh them out and stabilize them.

See the sample code answer tagged “FAQ(36)” (installed location information, or if you do not have the CTP installed, faq.cs) for a simple example.

Can I use the VS REPL outside of VS, either for quick standalone usage or hosting in an app? Can I host the REPL component in my application?

We do not have any current plan to enable the VS REPL outside of VS. It is dependent on Roslyn APIs that may only ship with VS. We are thinking of letting csi.exe drop into a REPL on a switch for quick executions of expressions or testing repro cases, but this REPL may be so bare bones that it doesn't support completion.

Do Roslyn Scripting APIs mean C# will be dynamically typed now, like a DLR language?

You do need to declare variables in the interactive/script semantics of C#, but you can use 'var' and 'dynamic'. All .NET code is strongly typed, even IronPython, but of course, IronPython is dynamically typed, not explicitly statically typed. The interactive/script semantics of C# let you use 'var' for top-level variables, which ultimately turn into fields on a Script class. You can use 'dynamic' as well to get dynamic or run-time binding of operations and method invocations, but Roslyn has not implemented ‘dynamic’ yet.

How do I supply host globals or context from the host to C# script code I'm loading with the Roslyn Scripting APIs?

When you create a Session object, or when you use certain overloads of Engine.Execute, you can supply an instance of a host object. When executing a supplied snippet of code, free identifiers in the code bind to public members of the host object.

See the sample code answer tagged “FAQ(36)” (installed location information, or if you do not have the CTP installed, faq.cs) for a simple example of sharing state across host globals and an example of sharing state in a session.

Is Roslyn Scripting a replacement for VBS, WSH, etc?

The answer depends on what you mean specifically. The answer is "yes" if your intent is to create a script that you can run on your machine or on another machine from cmd.exe as a simple program or utility. We imagine csi.exe is to enable you to write a simple program that you can just execute without all the ceremony (class, Main, project, etc.). We plan to ship csi.exe along with csc.exe, but that's not a promise at this point :-). Worse case, I'd expect csi.exe to be a sample. The intent of a default csi.rsp file is to enable to you author a script and have a friend or colleague run it with the same default execution context. You can also use a switch to not use the csi.rsp and then build up all your context with #r's in the script file.

The answer to this question is "no" if your intent is to have .csx files replace .vbs files from the standpoint of all the same WSH APIs, double click execution, etc. However, you could define an open verb for .csx files that invokes csi.exe for the double click, and to the extent WSH APIs are COM (and .NET had COM interop), you can use the same APIs. Emulating your VBS code may not be the best way to go given .NET APIs.

Miscellaneous Questions

What is elastic trivia?

Elastic trivia is usually in a manually constructed syntax tree to represent flexible whitespace elements. The trees returned by the parsers represent any whitespace literally as it was in the source code. Elastic whitespace lets generated trees suggest whitespace elements and ensure tokens are not immediately adjacent to each other. Formatters and other tree processing tools can freely substitute, lengthen, or change the elastic whitespace in any way without breaking fidelity with an original code source.

Why do some Syntax factories attach elastic trivia I didn't ask for?

Elastic whitespace denotes suggested whitespace, but rendering a tree to text may use zero or more whitespace where the elastic whitespace is. Some factories add elastic whitespace to help avoid accidently slamming two things together where you may require whitespace, or where a tool may want to apply options for how to format. Of course, a renderer needs to be smart and put some whitespace between some tokens, but it could, say, eliminate all whitespace around an open parenthesis depending on formatting options.

This elastic trivia will report that it has zero length because it potentially could be left out when mapping the tree to text.

You can take the result of the factory and remove the elastic trivia if you need to do so (though this would be very uncommon), for example:

```
Syntax.Token(SyntaxKind.SemicolonToken).WithLeadingTrivia().WithTrailingTrivia()
```

How do I format a tree or node to a textual representation?

See the sample code answer tagged "FAQ(35)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb) to see several formatting examples. This sample also include showing some services on IDocument, such as RemoveUnnecessaryImports() and SimplifyNames(). The sample "FAQ(26)" also shows SyntaxNode.Format(). Note, you need to use the Roslyn.Services namespace namespace and reference Roslyn.Services.dll and Roslyn.Services.{CSharp|VisualBasic}.dll to get these extension methods.

Can I use or interoperate with Powershell in the REPL?

Doug Finke has some cool demonstrations of invoking Powershell code from the C# REPL and writing cmdlets that enable using Powershell to query C# projects:

http://yfrog.com/g0u2obp

http://www.dougfinke.com/blog/index.php/2011/10/25/using-powershell-in-roslyns-c-interactive-window/

http://www.dougfinke.com/blog/index.php/2011/10/30/analyze-your-c-source-files-using-powershell-and-the-get-roslyninfo-cmdlet/

How can I use Roslyn in an MSBuild task and avoid metadata fetching and re-entrancy conflicts?

There is not a great answer for this currently. Roslyn uses MSBuild internally to construct workspaces, so when you try to construct them from within a Task, there are re-entrancy conflict issues. If it actually worked, there would also be an issue with a lot of duplicated effort and space. The right way would be to have a base RoslynTask that derived from Task and that used the MSBuild information already present. As a workaround for now, you can create a Roslyn Compilation from some of the info in the MSBuild Task and then use the compilation's code information. However, this still is not a complete solution since you likely want to affect the code being compiled, add files to the MSBuild targets, or remove them.

The following code is not in the samples/test project because it is not a great answer, but since the question comes up enough, this gives you some idea of what you need to do:

```
public class MyTask : Task {
    public override bool Execute() {
        var projectFileName = this.BuildEngine.ProjectFileOfTaskNode;
        var project = ProjectCollection.GlobalProjectCollection
                                       .GetLoadedProjects(projectFileName)
                                       .Single();
        var compilation =
            Compilation.Create(
                project.GetPropertyValue("AssemblyName"),
                syntaxTrees:
                project.GetItems("Compile")
                       .Select(c => SyntaxTree

.ParseCompilationUnit(c.EvaluatedInclude)),
                references:
                project.GetItems("Reference")
                       .Select(r => MetadataReference
                                       .Create(r.EvaluatedInclude)));
        // Now work with compilation ...
    }
}
```

Is there an end-to-end example on compiling a program to IL (Emit APIs)?

How can I capture IL, debug info, and doc comment outputs from a Compilation?

See the sample code answer tagged "FAQ(34)" (installed location information, or if you do not have the CTP installed, faq.cs or faq.vb). This sample includes an Execute() method definition, which compiles and executes a binary.

How do you use a VS plug-in written using Roslyn project?

The Visual Studio instances that run when you are debugging your Roslyn VS extension projects are launched under what we call the Roslyn "hive" (or regkey hierarchy where VS finds extensions). If you want to you can launch this instance of Visual Studio from the command-line by typing "devenv.exe /rootsuffix Roslyn".

Roslyn VS extensions need to access the Roslyn Language Services, and we only install the Roslyn Language Services under the Roslyn hive. Regular VS instances (started with no rootsuffix switch or some other rootsuffix) have the VS 2010 Language Services. This ensures that your regular code editing experience is not affected when you install the Roslyn CTP. However, this also means that Roslyn VS extensions won't be able to run under regular VS instances at the moment.

Eventually, once the Roslyn Language Services become the official Language Services inside VS, you would be able to run your extensions inside all instances of VS. In the interim, you can write extensions that only use the Roslyn Compiler APIs and no IDE APIs. Two examples of such extensions are the Paste As C#/VB extension and the Implement INotifyPropertyChanged extension.

How are AssemblyNameReference and AssemblyFileRefernce different?

When creating a Compilation, you need to supply references to compile against. You can do this in various forms. An easy way is to instantiate AssemblyFileReference on an expression such as "typeof(object).Assembly.Location". A more advanced way is to instantiate AssemblyNameReference, but though it takes names you could get from "typeof(object).Assembly.FullName", you must pass a MetadataReferenceResolver which can process the assembly name and resolve it to a file on disk or in the GAC.

How do I use MEF to import or export?

See Shyam's code answer here with links to working samples and docs:
http://social.msdn.microsoft.com/Forums/sk/roslyn/thread/b0a4d110-2ef0-4e94-8252-afe5b0562ec8

Is there an object model chart or type inheritance diagram of Roslyn types?

You can create a type inheritance diagram that you can zoom and search within. You need Visual Studio 2010 Ultimate, and the instructions for creating the diagram are in this post:

If you have Visual Studio 2010 Premium or Ultimate (or the Visual Studio 2012 RC Pro and up SKUs), then you can open these diagram files created for the Roslyn CTP2: C# or VB . These support zooming and searching, and when you click on a node, the viewer highlights links to help you see the relationships. You can also pivot the diagram or turn it into a butterfly graph to focus on one node.

You can also use the built-in Microsoft XPS viewer with these pictures: C# and VB. This viewer lets you search for a name and zoom in and out, but you cannot select nodes and highlight links, nor pivot the diagram.