C# Scripting

Filip W
Sonova
Switzerland

@filip_woj
strathweb.com
github.com/filipw

Plan

1. C# scripting?!
2. Executing / embedding in an app
3. Usage scenarios
4. How on Earth does the C# compiler do it?
5. Language services for scripts
6. Debugging


roslyn

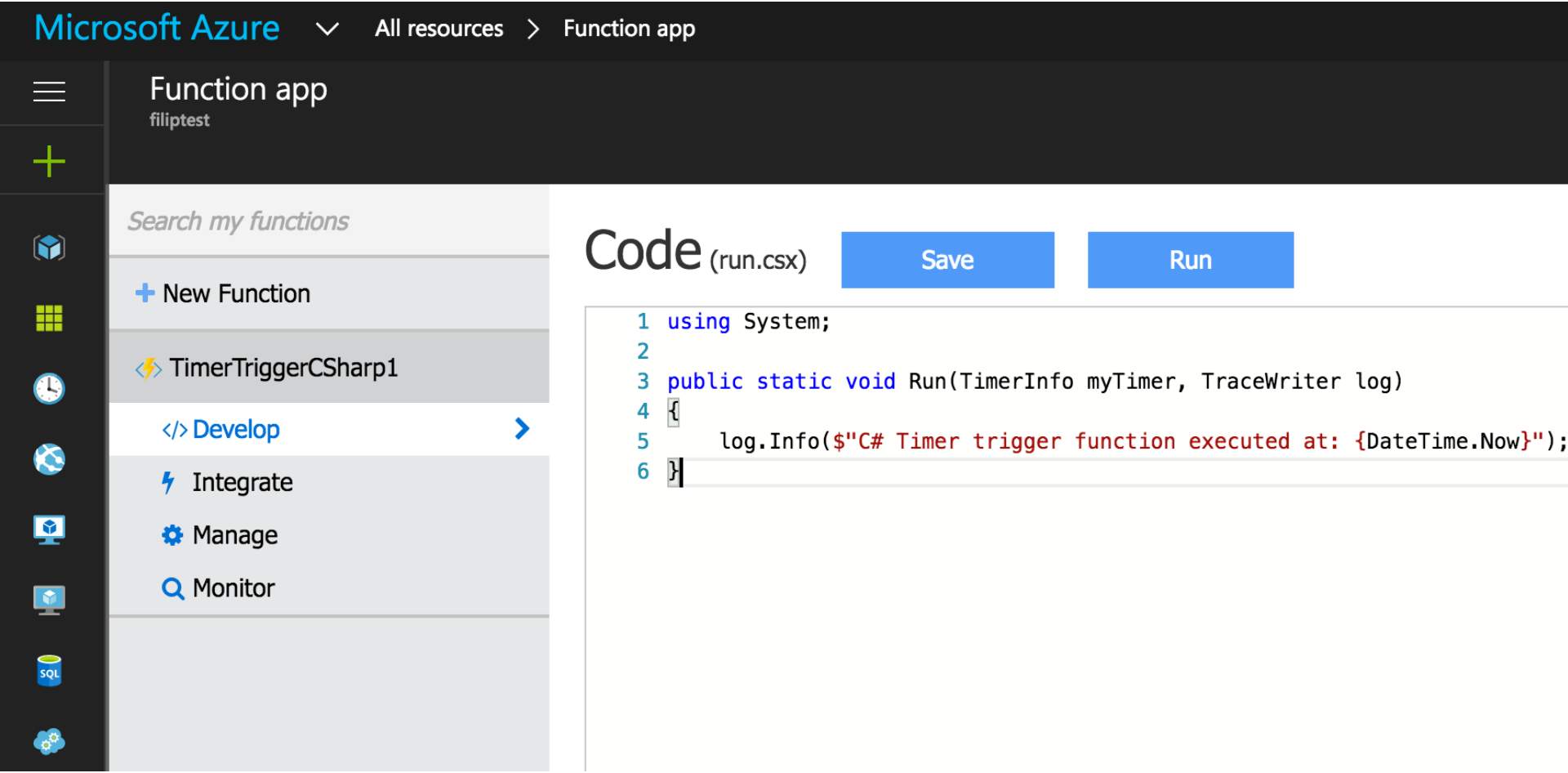
```
// no Program class
// no static void Main
// no namespace

// reference a script from a script
#load "anotherScript.csx"

// reference an existing assembly
#r "C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.5.2\System.Web.dll"

// import necessary namespaces
// System available implicitly
using System.Web;

// global functions
void PrintText(string name) {
    Console.WriteLine(name);
}

// global variables
var name = "filip";

// global statements
PrintText(name);
```

Script code semantics

- No top level class
- No script namespaces
- No *Main* method
- Loose functions / variables / statements allowed
- *#load* directive to load script from script
- *#r* directive to reference an assembly from script

Other considerations

- C# scripting is cross platform
- interactive (REPL) mode
- csi.exe runner comes with MSBuild tools
- embeddable in your own apps
(can marshal objects between host app and script)
- emitted IL is compatible with regular C# IL

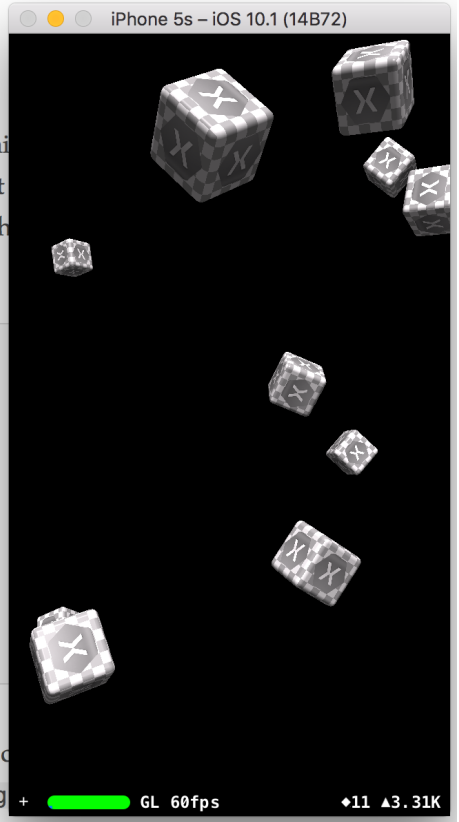

Microsoft Azure
All resources
Function app
Function app
filiptest
Search my functions
New Function
TimerTriggerCSharp1
Develop
Integrate
Manage
Monitor
Code (run.csx)
Save
Run
1 using System;
2
3 public static void Run(TimerInfo myTimer, TraceWriter log)
4 {
5 log.Info($"C# Timer trigger function executed at: {DateTime.Now}");
6 }

Our scene graph is empty, so let's set the stage:

Lights!

There are a variety of SCNLightType values. We need at least one to see anythi
we'll create an "omni" light that emits light in all directions (like a naked light
"ambient" light that illuminates all objects in the scene from all directions with
intensity:

```
var lightNode = new SCNNode();
lightNode.Light = new SCNLight ();
lightNode.Light.LightType = SCNLightType.Omni;
lightNode.Position = new SCNVector3 (0.0F, 10.0F, 10.0F);
scene.RootNode.AddChildNode (lightNode);

var ambientLightNode = new SCNNode ();
ambientLightNode.Light = new SCNLight ();
ambientLightNode.Light.LightType = SCNLightType.Ambient;
ambientLightNode.Light.Color = UIColor.DarkGray;
scene.RootNode.AddChildNode (ambientLightNode);
```



The above code shows a typical pattern for working with SceneKit. First, you c
`SCNNode`. Second, you set properties of that node such as it's `Light` and `Lig`
properties. The `Position` property locates the `SCNNode` in 3D space *relative to its parent node's* `Position`, `Rotation`, and `Scale` properties. Finally, you add the newly-created `SCNNode` as a child of an existing `SCNNode`.

Miguel de Icaza
@migueldeicaza

Following

C# scripting coming to Minecraft

Searge @SeargeDP
At Minecon I talked about how we will bring C# plugins to Minecraft with the help of Xamarin. Thanks to @migueldeicaza and his team for Mono
twitter.com/SeargeDP/statu…

RETWEETS 114 LIKES 107

11:59 AM - 7 Oct 2016

114 107

Slides are boring – code time

C# scripting use cases

- lightweight programs
- experiments / exploration
- automation
- application extensibility
- dynamic configuration
- code generation (T4-like capabilities)

How does it work?

```
class Foo {}

void SayHi(string msg)
{
    Console.WriteLine(msg);
}

var message = "hello!";
var foo = new Foo();

SayHi(message + " " + Text);
```

becomes (roughly) …

```
public sealed class Submission#0
{
    public string message;
    public Submission#0.Foo foo;
    public MyHost <host-object>;
    public class Foo {}

    public void SayHi(string msg) {
        Console.WriteLine(msg);
    }

    public Submission#0(object[] submissionArray) {
        submissionArray[1] = this;
        this.<host-object> = (MyHost)submissionArray[0];
    }

    public static Task<object> <Factory>(object[] submissionArray) {
        return new Submission#0(submissionArray).<Initialize>();
    }

    // returns a Task built up from the nested <<Initialize>> state machine
    public Task<object> <Initialize>() {}

    // shortened for brevity
    public sealed class <<Initialize>>d__0 : IAsyncStateMachine {
        public AsyncTaskMethodBuilder<object> <>t__builder;
        public Submission#0 <>4__this;

        void IAsyncStateMachine.MoveNext() {
            this.<>4__this.message = "hello!";
            this.<>4__this.SayHi(this.<>4__this.message + " " +
                this.<>4__this.<host-object>.Text;
            this.<>4__this.foo = new Submission#0.Foo();
        }
    }
}
```

- code is wrapped into a *Submission* class
- global variables/methods -> public instance members
- host object -> public field
- classes -> nested classes
- loose code is wrapped in an *<Initialize>* class
- *Static <Factory>* method is entry point

C# scripting is a first class compiler citizen

Some Roslyn features

- ability to parse and understand script syntax trees
  * *SourceCodeKind.Script*
- create script compilations
  * *CSharpCompilation.CreateScriptCompilation()*
- provide script language services
  * *ProjectInfo* knows host object *Type* & global usings


WITH CSX WE CAN USE C#
FOR SCRI...
NO INTELLISENSE?!

MessageSecurityVersion.WSSecurity11WSTrustFebruary2005WSSecureConversationFebruary2005WSSecurityPolicy11BasicSecurityProfile10 Property

Gets the message security version that requires the Basic Security Profile based on WS-Security 1.1, WS-Trust of February 2005, WS-SecureConversation of February 2005 and WS-SecurityPolicy 1.1 security specifications.

Namespace: System.ServiceModel
Assemblies: System.ServiceModel.Security (in System.ServiceModel.Security.dll)
System.ServiceModel (in System.ServiceModel.dll)

Syntax

C# | C++ | F# | JScript | VB

```
public static MessageSecurityVersion WSSecurity11WSTrustFebruary2005WSSecureConversationFebruary2005WSSecurityPolicy11BasicSecurityProfile10 { get; }
```

Property Value
Type: System.ServiceModel.MessageSecurityVersion
The MessageSecurityVersion object that requires the Basic Security Profile based on WS-Security 1.1, WS-Trust of February 2005, WS-SecureConversation of February 2005 and WS-SecurityPolicy 1.1 security specifications.

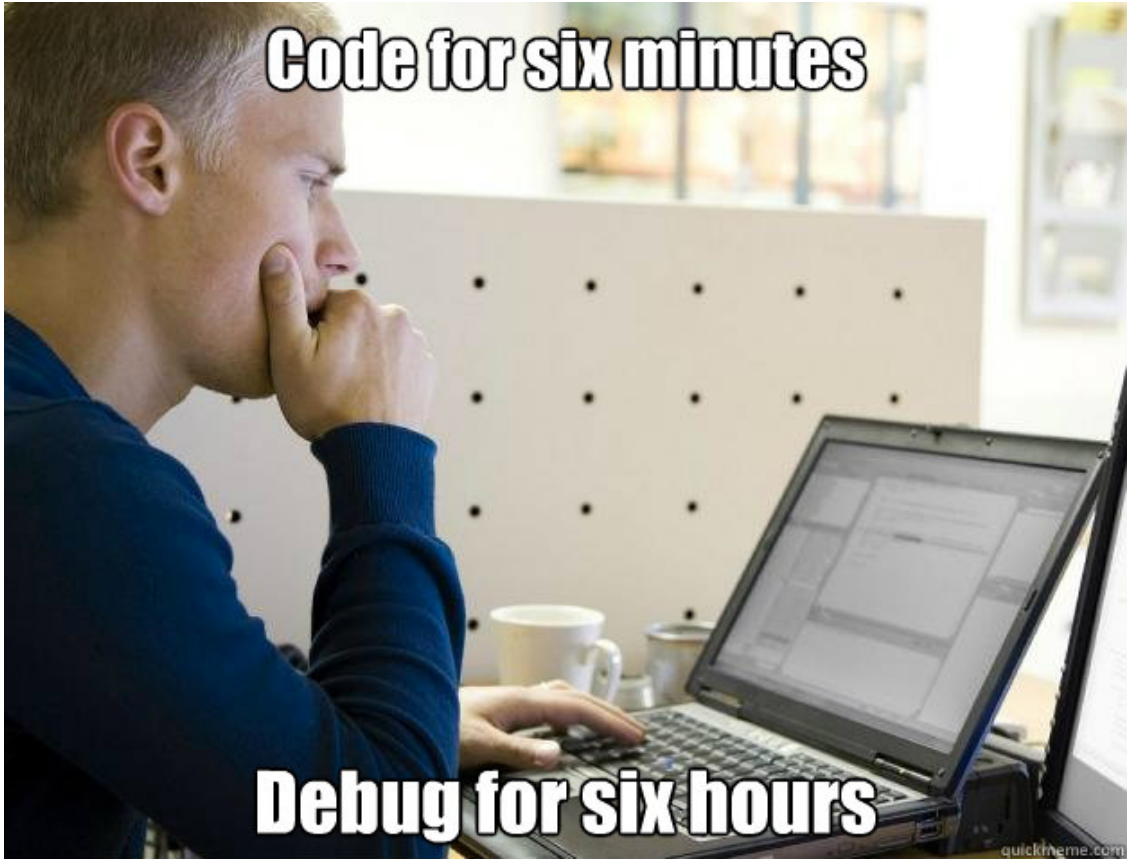
Remarks

This static property returns an instance of this class. The instance of this class is used as parameters in the `Create…CertificateBindingElement` method calls on the SecurityBindingElement class.

Version Information

.NET Framework
Supported in: 4.6, 4.5, 4, 3.5, 3.0
.NET Framework Client Profile
Supported in: 4, 3.5 SP1

Slides are boring – code time


Code for six minutes
Debug for six hours
quickmeme.com


made with dwigif.appspot.com

The end