ClearScript FAQtorial

1. What's a FAQtorial?

It's a tutorial in the form of questions and answers. It may look like a FAQ list, but it's really a guide.

2. OK, so what's ClearScript?

ClearScript is a library that allows you to add scripting to your .NET applications. It supports JScript and VBScript out of the box and in theory can work with other Windows Script engines.

New! ClearScript 5 supports the V8 high-performance open-source JavaScript engine. Unlike Windows Script engine instances, V8 instances have no thread affinity and are suitable for server-side asynchronous scripting.

3. Hasn't that been done before?

Perhaps, but sometimes it's fun to start something from scratch and see how far you can take it.

4. Why would I want to add scripting to my application anyway?

There are many possible reasons. Sophisticated productivity applications often support script-based customization and extension. The ability to automate an application may be desirable for testing purposes or for running complex jobs. Another goal might be to process active content that includes embedded scripts, such as web pages.

5. Doesn't .NET have its own scripting technologies?

Yes, but they're designed for a different purpose. They typically introduce numerous language extensions in order to "retarget" standard script languages for full-blown .NET development. That is *not* ClearScript's goal. ClearScript focuses on adding standards-based scripting to *existing* .NET applications with minimal effort.

6. OK, but if it's been done before, what's so special about ClearScript?

ClearScript is all about simplicity. It's one line of code to instantiate a script engine, another to expose a host resource for scripting, and one more to execute a script. You don't worry about implementing hosting interfaces, creating scriptable proxies, decorating class members, etc.

Because it aims to make existing applications scriptable without modification or special coding, ClearScript works hard to give script code the ability to access a long list of .NET features – constructors, methods, properties, fields, events, indexers, extension methods, nested types, output parameters, statics, generics, enums, delegates, etc. It does this without introducing special syntax or requiring script language extensions of any kind.

ClearScript also provides streamlined host-to-script support, allowing the host to invoke script functions and access script objects directly. In addition, it allows script-enabled applications to be debugged in script mode.

New! ClearScript 5 supports simultaneous debugging of script, managed, and native code for applications that use its new V8-based JavaScript engine.

7. Fine, so how do I instantiate a script engine with ClearScript?

You simply "new up" a class instance:

```
Host - C#
using Microsoft.ClearScript.V8;
…
var engine = new V8ScriptEngine();
```

At this point the script engine contains only its built-in objects and functions and has no access to the host application.

8. That's not very useful; how do I expose my host API for scripting?

You make one method call to expose a host object or type:

```
Host - C#
using System;
…
engine.AddHostObject("uri", new Uri("http://www.example.com"));
engine.AddHostType("Console", typeof(Console));
```

The script engine now contains two root-level objects – a *host object* and a *host type* – that provide access to the corresponding host resources.

9. What can script code do with a host object?

It can access the object's public members directly:

```
Script - JavaScript
var uriQuery = uri.Query;
var uriString = uri.ToString();
```

10. What can script code do with a host type?

It can access the type's static members and nested types directly:

```
Script - JavaScript
Console.WriteLine('Downloading from {0}...', uri);
Console.Title = 'Please wait...';
```

Host types can also be used to invoke constructors and specify generic type arguments. There'll be more on that later.

11. Once my host API is exposed, how do I actually run a script?

Use one of these methods:

```
Host - C#
engine.Execute("var uriQuery = uri.Query;");
var result = engine.Evaluate("Math.sqrt(Math.PI)");
```

12. Why have two different methods for running scripts?

Some script languages have ambiguous syntactic elements whose meaning depends on whether they're part of a statement or an expression. For example, in VBScript a single equals sign is used for both assignment (statement)

and equality testing (expression). To resolve this ambiguity the host must establish the syntactic context for script execution. **`Execute`** interprets script code in a statement context, while **`Evaluate`** treats code as an expression.

13. You've covered host methods and properties. What about fields?

To script code, the public fields of a host object or type look exactly like properties.

14. How do I call generic methods?

ClearScript supports C#-like type inference, so you can call most generic methods just as you'd call them from C#. However, some generic methods require explicit type arguments. To call such a method from script code, place the required number of host types at the beginning of the argument list. For example, suppose the host exposes the following types:

```
Host – C#
using System;
using System.Linq;
…
engine.AddHostType("Int32", typeof(int));
engine.AddHostType("Enumerable", typeof(Enumerable));
```

Here's how to call the **`Enumerable.Empty`** generic method from script code:

```
Script - JavaScript
var empty = Enumerable.Empty(Int32);
```

15. What about extension methods?

Extension methods are available if the type that implements them has been exposed. Continuing the previous example, here's script code that calls the **`Enumerable.ToArray`** extension method:

```
Script - JavaScript
var numbers = Enumerable.Range(0, 10).ToArray();
```

Note that **`Enumerable.ToArray`** is a generic extension method whose type argument is inferred from its "this" argument. As you can see, ClearScript allows you to call it using the same convenient syntax as C#.

16. Can I handle an event using a script function?

Yes. Here's how to connect a host event source to a script handler function:

```
Script - JavaScript
var connection = Console.CancelKeyPress.connect(function (sender, args) {
    Console.WriteLine('{0} pressed!', args.SpecialKey);
});
```

To break the connection, do this:

```
Script - JavaScript
connection.disconnect();
```

17. How do I create an instance of a host type from script code?

Script code cannot instantiate a host type directly. However, ClearScript provides a set of utility functions that includes this capability:

```
Host – C#
using System;
using Microsoft.ClearScript;
…
engine.AddHostObject("host", new HostFunctions());
engine.AddHostType("Uri", typeof(Uri));
```

Now you can create **System.Uri** class instances from script code:

```
Script - JavaScript
var newUri = host.newObj(Uri, 'http://www.example.com');
```

Note that **newObj** is a generic method that requires an explicit type argument. Its first argument (the type argument) is the host type to be instantiated. The rest of the arguments are passed to the constructor that matches the argument signature.

New! ClearScript 5 supports simplified instantiation syntax for JavaScript:

```
Script - JavaScript
var newUri = new Uri('http://www.example.com');
```

18. What about an array?

There's a utility function for creating host arrays, too:

```
Script - JavaScript
var uriArray = host.newArr(Uri, 10);
```

19. Can I access host array elements by index?

That depends on your script language. However, if a host object implements **System.Collections.IList**, ClearScript exposes read/write properties with numeric names that match the object's valid indices:

```
Script - JavaScript
var firstUri = uriArray[0];
uriArray[1] = firstUri;
```

20. Can I call methods with output parameters from script code?

Yes. Suppose that a dictionary is exposed for scripting:

```
Host – C#
using System.Collections.Generic;
…
engine.AddHostObject("dict", new Dictionary<string, Uri>());
```

You can use a *host variable* to invoke the **Dictionary.TryGetValue** method from script code:

```
Script - JavaScript
var uriVar = host.newVar(Uri);
if (dict.TryGetValue('key', uriVar.out)) {
    Console.WriteLine('Found Uri value: {0}.', uriVar);
}
```

Host variables support two special properties in addition to **out**. The first is **ref**, which is similar to **out** but compatible with **ref** parameters. The second is **value**, which provides read/write access to the value stored within the variable. If you're passing a variable into an input parameter, invoking **value** is unnecessary.

21. Can I route a callback to a script function?

Yes, by creating a delegate that invokes your script function. Consider this code:

```
Host – C#
using System.Threading;
…
engine.AddHostType("Timer", typeof(Timer));
engine.AddHostType("TimerCallback", typeof(TimerCallback));
```

You can now create timers and handle their callbacks from script code:

```
Script - JavaScript
var callback = host.del(TimerCallback, function (state) {
    Console.WriteLine('Timer fired: {0}.', state);
});
var timer = host.newObj(Timer, callback, 'foo', 5000, 5000);
```

In addition to **del**, ClearScript provides **proc** and **func**; these functions create specialized delegates compatible with **System.Action** and **System.Func** respectively.

New! ClearScript 5 supports simplified delegate creation syntax for JavaScript:

```
Script - JavaScript
var callback =  new TimerCallback(function (state) {
    Console.WriteLine('Timer fired: {0}.', state);
});
```

22. Can I expose many host types in one step?

Yes. You can use a *host type collection* to expose all the types defined in one or more assemblies:

```
Host - C#
using Microsoft.ClearScript;
…
var typeCollection = new HostTypeCollection("mscorlib", "System", "System.Core");
engine.AddHostObject("clr", typeCollection);
```

Now all the types defined in the standard **mscorlib**, **System**, and **System.Core** assemblies are exposed. Host type collections are hierarchical data structures where leaf nodes represent host types and parent nodes represent namespaces:

```
Script - JavaScript
var guid = clr.System.Guid.NewGuid();
var today = clr.System.DayOfWeek.Friday;
```

Note that, unlike C# types and namespaces, the nodes of a host type collection are objects. A script can copy them to the root level for more convenient access:

```
Script - JavaScript
var System = clr.System;
var Guid = System.Guid;
var myGuid = Guid.NewGuid();
```

23. What if a host type refers to an open generic type?

Such a host type can be invoked with type arguments to produce a closed generic type:

```
Script - JavaScript
var Dictionary = System.Collections.Generic.Dictionary;
var myDictionary = host.newObj(Dictionary(System.String, System.Int32));
myDictionary.Add('foo', 123);
```

24. Can I import host types from script code?

Not without the host's explicit permission:

```
Host – C#
using Microsoft.ClearScript;
…
engine.AddHostObject("xHost", new ExtendedHostFunctions());
```

By exposing an instance of **`ExtendedHostFunctions`**, the host hands over the keys to the kingdom:

```
Script - JavaScript
var List = xHost.type('System.Collections.Generic.List');
var DayOfWeek = xHost.type('System.DayOfWeek');
var week = xHost.newObj(List(DayOfWeek), 7);
week.Add(DayOfWeek.Sunday);
```

You can even import entire assemblies:

```
Script - JavaScript
var clr = xHost.lib('mscorlib', 'System', 'System.Core');
var Dictionary = clr.System.Collections.Generic.Dictionary;
```

25. Can the host call script functions and access script objects directly?

Yes. ClearScript provides a dynamic property named **`Script`** that represents a script engine's root-level namespace. Consider the following script:

```
Script - JavaScript
function myFunc(x, y, z) {
    return { xValue: x, yValue: y, zValue: z };
}
```

After executing this script, the host can directly invoke **myFunc** and examine the properties of the object it returns:

```
Host – C#
var yValue = engine.Script.myFunc(1, 2.0, "three").yValue;
```

26. How do I debug script code?

If you're using ClearScript with JScript, VBScript, or another Windows Script engine, you can use Visual Studio to debug your application in script mode. If you're using ClearScript's new V8-based JavaScript engine, your best option is Eclipse combined with Google Chrome Developer Tools for Java. Best of all is that you can attach both Visual Studio and Eclipse to the same process for simultaneous debugging of script, managed, and native code. See the **ReadMe.txt** file for the latest information about using ClearScript with V8.

27. How can I learn more?

Take a look at the ClearScript API reference that accompanies this FAQtorial. Also check out the **ClearScriptConsole** and **ClearScriptBenchmarks** test programs.