Server-Side JavaScript in 1997

Adrian Kosmaczewski

2021-01-29

Back in 1997 I was earning some cash writing Active Server Pages in that mutant programming language called VBScript. On the other hand, Microsoft had reverse-engineered the JavaScript compiler inside Netscape Navigator (there was no spec, after all), and they created a dialect of it called JScript (they could not name it JavaScript because lawyers) and made it compatible with their COM+ runtime model.

These were the 90s; the future was COM+ components using `IUnknown` and `IDispatch` interfaces, so that you could create applications using COM components created in C++, stitched together with a scripting language, like JScript of VBScript. A bit like how Objective-C wraps low-level C libraries, a bit like how Swift wraps low-level Objective-C frameworks. Always the same idea.

Of course the future turned out to be .NET, not COM+, and now I think their next platform should be called ORG just for the sake of playing with domain names.

I know, weird times and even weirder names. But the truth is that in 1998 I was literally writing server-side JavaScript, but as you can imagine it was neither convenient nor performant nor hype. Actually people hated JavaScript vehemently.

It is not like there were many choices. There was Cold Fusion, but programming with HTML tags still rings creepy to me. PHP was not yet an option. WebObjects might have been one, albeit a much more expensive one. Like, 50'000 USD *per server license*. And we did not know anything about Objective-C. Lisp might have been another, but we knew even less about it.

So VBScript it was.

Why did people hate JavaScript so much? That is a question that I have asked myself quite a few times during the past 22 years. I think it has to do with broken promises. JavaScript looks like Java, it even has a name that sounds like Java, it has curly brackets and semicolons like Java, yet the `this` pointer usually points somewhere else than where one expects. Hence people gets pissed off, and they rejected all contact with the language for as long as they could.

However, life is a bitch, and JavaScript being the *only* programming language to write stuff running on a web browser, that meant sooner or later you would have to deal with it, anyway, no matter how much you hated it. We had to wait until 2001 for Douglas Crockford to be the first person to actually understand JavaScript and then expose its good parts seven years later in a now absolutely classic book.

So people came up with many strategies to cope with the pain. The earliest attempts around 2010 were CoffeeScript (targeting Python lovers) and Cappuccino (for the Objective-C demographic). Microsoft's (once again) TypeScript is one of the latest and, at the time of this writing, one of the most popular options these days, together with Elm, Kotlin, ClojureScript, LiveScript, Amber (for Smalltalk folks), Dart, Fable (for F# aficionados), Opal (for Ruby fanatics), and there are new ones popping every week.

Clearly nobody wants to deal with plain JavaScript. I understand them. But I digress.

On the server side, however, ASP had a surprisingly simple programming model. Everything was quite procedural, one mixed and matched HTML and code as you would in PHP nowadays, and you could use five big global objects, named after their respective classes, to make stuff happen: `Application`, `Request`, `Response`, `Server`, and `Session`. You could put global state at application or session level, the runtime provided basic I/O functionality, and lots of functions to perform whatever task you wanted to do.

And if you needed something fancier, you could always buy a COM+ component from a third party, install it in your server and instantiate it to perform some tasks, like sending e-mail from the server, which was not available off-the-box in the ASP runtime. You read right.

VBScript did not allow you to create classes, at least not in the sense that you might expect; there was a `Class` keyword, yes, but it merely created in-memory structures, which for reasons nobody understood could not be stored neither on a Session nor on an Application object, and they were as such only useful as parameter or return types for functions. Quite useless as a matter of fact, *mais c'est la vie.*

To be fair, there was a COM object implementing a simple hashtable object one could use from within VBScript, giving you something like this:

```
Function DicDemo
   Dim a, d, i, s   ' Create some variables.
   Set d = CreateObject("Scripting.Dictionary")
   d.Add "a", "Athens"   ' Add some keys and items.
   d.Add "b", "Belgrade"
   d.Add "c", "Cairo"
   a = d.Items   ' Get the items.
   For i = 0 To d.Count -1 ' Iterate the array.
```

```
        s = s & a(i) & "<BR>" ' Create return string.
    Next
    DicDemo = s
End Function
```

As you might expect, the same objects were available from JavaScript; the difference was that instead of using `CreateObject()` you would use `new ActiveXObject()` instead, passing the name of the COM class as a parameter. All very dynamic.

```
function ItemsDemo()
{
    var a, d, i, s;                     // Create some variables.
    d = new ActiveXObject("Scripting.Dictionary");
    d.Add ("a", "Athens");                  // Add some keys and items.
    d.Add ("b", "Belgrade");
    d.Add ("c", "Cairo");
    a = (new VBArray(d.Items())).toArray();    // Get the items.
    s = "";
    for (i in a)                       // Iterate the dictionary.
    {
        s += a[i] + "<br>";
    }
    return(s);                          // Return the results.
}
```

I hope you have not missed that `new VBArray()` thing in line 8.

The truth is that even if we could have used JavaScript both on the client and the server side, we wrote most of the backend in VBScript. The thing is, many features in the ASP runtime did not work correctly on JScript, like for example disconnected `ADODB.RecordSet` objects. Well, theoretically you could make them work, but the page that explained how no longer exists, and it only worked in later versions of IIS and ASP.

Also, we were early adopters of the "Internet-Based Programming" methodology, way before Stack Overflow existed, waaaaay before Joel Spolsky started blogging. Since most of the snippets you found on the web were in VBScript and we were really lazy developers, well, the backend ended up in VBScript and that was it.

Somebody even came up with a tool for unit tests in VBScript in 2005, but by that time I was already doing something else.