



Unobtrusive Scripting

Table Of Contents

- [The Old Way](#)
- [Be Unobtrusive](#)

The Old Way

When you want to add some extra functionality to the webpage such as a back button or link or providing instant user interaction beyond forms, you would use a `<script language="JavaScript"></script>` or `<script language="JavaScript" src="path/to/doi.js"></script>` element using JavaScript or Microsoft's JScript.

Then of course you had to add a `<noscript></noscript>` element as an afterthought to address those that had scripting turned off or not supported.

Now these days the `language` attribute is depreciated or completely dropped in favour of the global standard of the `type` attribute providing a MIME Media Type to identify the scripting language: `<script type="text/javascript"></script>` or `<script type="text/javascript" src="path/to/doi.js"></script>`. Plus the Web Content Accessibility Guidelines (WCAG) 1.0 dictate the use of noscript elements for scripts. But these guidelines were published in 1999! Many features of web browsers, assistive technologies such as screen readers and web development techniques have very much improved since WCAG1. Version 2 of WCAG has been in development for most of this time too.

Be Unobtrusive

As a rule of thumb, the website must be navigable and information on a webpage must be available and readable without any enhancement or extra technology. So Markup such as HTML (HyperText Markup Language), XML (eXtensible Markup Language) based and HTML5 allow the baseline of a webpage: to structure the information. Styling and scripting are secondary features to enhance the web experience. Styling such as using CSS (Cascade StyleSheets) enhance the presentation of the information and provides some functionality to enhance the interaction between user and the information.

Scripting such as using JavaScript enhances the interaction even further. But if the information needs styling and or scripting to access and read it then that information has no business existing.

Most of the enhanced interaction with the information provided by scripting of course can't be provided by Markup or CSS. (Otherwise why would you be using scripting when you could just use the Markup or CSS instead.) The non-scripting version should provide a brief description of what the scripted version would provide or specifically in case of a javascript back link you would have a placeholder for the scripted link (if the link requires scripting to work and scripting is disabled, then the link would not work so it is best to not have a link present so people can't click the link and wonder why it doesn't work).

So the non-scripting version, if any, should be part of the normal flow of the document, instead of wrapped up in a noscript element. The script's job is to add, append, prepend or overwrite the direct non-scripted content with the

Unobtrusive Scripting - Legend Scrolls

scriptable content.

If scripting is disabled or not supported then the non-scripted content in the normal flow of the document is provided by default and the script element and its contents are not processed – thus the `<noscript></noscript>` element is a waste of markup. If scripting is supported and enabled then the non-scripted content is there by default while the script is processed and due to the scripting is replaced or augmented by the result of the script and again `<noscript></noscript>` is not needed.

Back link:

```
<p id="backLink"></p>
<script type="text/javascript" src="path/to/backlink.js"></script>
```

In backlink.js:

```
var bl = document.getElementById("backLink");
bl.innerHTML = "<a href=\"javascript:window.history.back()\">&lt; Back to
previous page</a>";
```

End.

Unobtrusive Scripting - Legend Scrolls

This quick tutorial is available online at:

<http://www.legendscrolls.co.uk/articles/unobtrusivescripting/>

Author: Peter Davison from [Legend Scrolls](#).

Copyright ©2008 Legend Scrolls and Peter Davison.

All rights reserved.