Embedding external script

Where it is desirable to create a single portable HTML document its functionality can be provided by a **<script>** element within the document's head section, as in the previous example, and styling can be provided by a **<style>** element within the head section.

Where portability is of no importance greater efficiency can be achieved by creating external script and style files. For instance, all the examples in this chapter employ the same single style file to create the Web 2.0 look around the panel element. Similarly all HTML files throughout a website could therefore employ a single script file to embed JavaScript functionality in each web page. Often the JavaScript file may be referred to as a "library" because it contains a series on behavioral functions which can be called from any page on that website.

Embedding an external JavaScript file in the head section of a HTML document requires a **src** attribute be added to the usual **<script>** tag to specify the path to the script file. Where the script file is located in the same directory as the HTML document this merely needs to specify its file name and file extension – typically JavaScript files are given a ".js" file extension. For example, you can embed a local JavaScript file named "local.js" like this:

```
<script type="text/javascript" src="local.js"> </script>
```

The separation of structure (HTML), presentation (Cascading Style Sheets), and behavior (JavaScript), is recommended by the WorldWideWeb Consortium (W3C) as it makes site maintenance much simpler and each HTML document much cleaner – and so easier to validate.

Using HTML event attributes, such as **onload**, **onmouseover**, etc., to specify behavior continues to intrude on the structural nature of the HTML elements and is not in the spirit of the W3C recommendation. It is better to specify the behaviors in JavaScript code contained in an external file so the HTML document contains only structural elements, embedding behaviors and styles from elements in the head section specifying their file locations. The technique of completely separating structure and behavior in this way creates unobtrusive JavaScript, which is considered to be "best practise" and is employed throughout the rest of this book.

The W3C is the recognized body that oversees standards on the web. See the latest developments on their informative website at **www.w3.org**.

Don't forget

Remember to add the closing **</script>** tag. It is required even though the element is empty.

1. Create a HTML document then add a **<div>** element to its body section, with an **id** attribute value of "panel", and containing alternative text for when JavaScript is absent

```
<div id="panel"><noscript>
  <div>! JavaScript is Not Enabled.</div></noscript>
</div>
```

external.html

2. In the head section of the HTML document, insert an element to embed an external JavaScript file

```
<script type="text/javascript" src="external.js"></script>
```

3. Open a plain text editor, like Windows Notepad, and add an "init" function to write content in the panel and to display a message dialog box

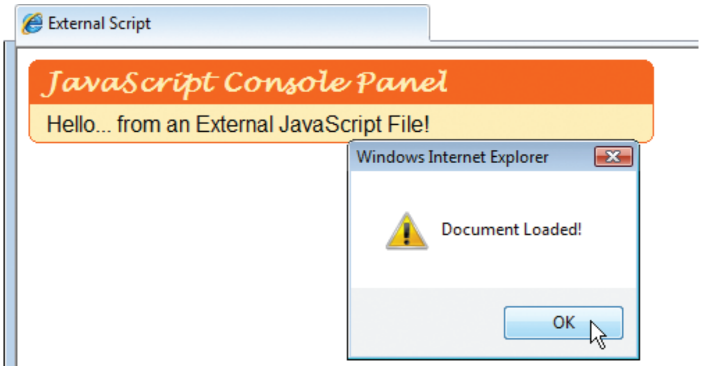
```
function init()
{
  document.getElementById( "panel" ).innerHTML=
          "Hello... from an External JavaScript File!" ;
  window.alert( "Document Loaded!" ) ;
}
```

external.js

4. After the function block, add a statement to call the function whenever the HTML document gets loaded

```
window.onload=init ;
```

5. Save the script alongside the HTML document then open the page in your browser to see the text and dialog



External Script

JavaScript Console Panel

Hello... from an External JavaScript File!

Windows Internet Explorer

Document Loaded!

In this example the function name, without parentheses, is assigned to the **onload** property of the **window** DOM object.

Beware

An error will occur if you include parentheses when assigning a function to the **window.onload** property – just assign its name.