**JavaScript and HTML**

You can think of JavaScript as an extension to HTML; an add-on, if you will.

Here's how it works. HTML tags create objects; JavaScript lets you manipulate those objects. For example, you use the HTML <BODY> and </BODY> tags to create a Web page, or *document.* As shown in Table 1, after that document is created, you can interact with it by using JavaScript. For example, you can use a special JavaScript construct called the onLoad *event handler* to trigger an action — play a little welcoming tune, perhaps — when the document is loaded onto a Web browser. Examples of other HTML objects that you interact with using JavaScript include windows, text fields, images, and embedded Java applets.

**Table 1: Creating and Working with Objects**

|  |  |  |
| --- | --- | --- |
| ***Object*** | ***HTML Tag*** | ***JavaScript*** |
| Web page | <BODY> . . . </BODY> | document |
| Image | <IMG NAME="myImage"> | document.myImage |
| HTML form | <FORM name="myForm"> . . . </FORM> | document.myForm |
| Button | <INPUT TYPE="button" NAME="myButton"> | document.myForm.myButton |

To add JavaScript to a Web page, all you have to do is embed JavaScript code in an HTML file. Below the line in which you embed the JavaScript code, you can reference, or *call,* that JavaScript code in response to an event handler or an HTML link.

You have two choices when it comes to embedding JavaScript code in an HTML file:

* **You can use the <SCRIPT> and </SCRIPT> tags to include JavaScript code directly into an HTML file.**

In the example below, a JavaScript function called processOrder() is defined at the top of the HTML file. Further down in the HTML file, the JavaScript function is associated with an event handler — specifically, the processOrder button's onClick event handler. (In other words, the JavaScript code contained in the processOrder() function runs when a user clicks the processOrder button.)

<HTML>  
<HEAD>  
<SCRIPT LANGUAGE="JavaScript">  
// JavaScript statements go here  
function processOrder() {  
// More JavaScript statements go here  
}  
</SCRIPT>  
</HEAD>  
<BODY>  
<FORM NAME="myForm">  
<INPUT TYPE="button" NAME="processOrder" VALUE="Click to process your order" onClick="processOrder();">  
...  
</HTML>

* **You can use the <SCRIPT> and </SCRIPT> tags to include a separate, external JavaScript file (a file containing only JavaScript statements and bearing a .js extension) into an HTML file.**

In the example below, the JavaScript processOrder() function is defined in the myJSfile.js file. The function is triggered, or *called,* when the user clicks the Click Here to Process Your Order link.

<HTML>  
<HEAD>  
<SCRIPT LANGUAGE="JavaScript" SRC="myJSfile.js">  
</SCRIPT>  
</HEAD>  
<BODY>  
<A HREF="javascript:processOrder();">Click here to process your order.</A>  
...  
</BODY>  
</HTML>

Because Web pages aren't made of HTML alone, JavaScript provides access to more than just HTML objects. JavaScript also provides access to browser- and platform-specific objects. Browser plug-ins (such as RealPlayer and Adobe Acrobat), the name and version of a particular viewer's browser, and the current date are all examples of non-HTML objects that you can work with by using JavaScript.

Together, all the objects that make up a Web site — HTML objects, browser- and platform-related objects, and special objects built right into the JavaScript language — are known as the *document object model* (DOM).