



## Capitolo 3

# Il primo script in JavaScript

- 3.1 **Gli strumenti software**
- 3.2 **Impostazione dell'ambiente di authoring**
- 3.3 **Cosa farà il primo script**
- 3.4 **Inserimento del primo script**
- 3.5 **Analisi dello script**
- 3.6 **Un po' di divertimento**

In questo capitolo verrà impostato un ambiente per la scrittura e per la visualizzazione in anteprima degli script, poi verrà realizzato un semplice script i cui risultati possono essere visti in un browser compatibile con JavaScript.

A causa delle differenze di comportamento tra i vari sistemi operativi, saranno presentati i dettagli degli ambienti per due varianti comuni: Windows (da 95 a XP) e Mac OS X. Per la maggior parte l'esperienza di authoring con JavaScript è la stessa, indipendentemente dal sistema operativo utilizzato (compresi Linux e UNIX). Anche se ci possono essere leggere differenze nei tipi di carattere a seconda del browser e del sistema operativo, le informazioni restano le stesse. La maggior parte delle illustrazioni sull'output nei browser in questo libro è realizzata con la versione Windows XP di Internet Explorer 6. Se viene utilizzato un altro browser o un'altra versione non occorre preoccuparsi se il risultato non è perfettamente identico a quello delle illustrazioni presentate nel libro.

### 3.1 Gli strumenti software

Il modo migliore per imparare JavaScript è digitare il codice HTML e di scripting dei documenti con un editor di testi. La scelta dell'editor di testi è personale, sebbene nel prossimo paragrafo siano presentate alcune linee guida per la scelta del programma.

#### Scelta di un editor di testi

Per gli scopi di apprendimento di JavaScript in questo libro, per il momento è meglio evitare gli strumenti di authoring Web WYSIWYG (*What You See Is What You Get*, ciò che si vede è ciò che si ottiene) quali FrontPage e



Dreamweaver. Questi strumenti diventeranno sicuramente utili in un secondo momento, quando potranno essere utilizzati in modo produttivo per modellare gran parte del contenuto e del layout. Tuttavia, gli esempi in questo libro si concentrano più sul contenuto degli script (che comunque è necessario scrivere), quindi non sarà necessario digitare molto codice HTML. I file di tutti i listati completi delle pagine Web (eccetto per i capitoli del tutorial) sono presenti anche sul sito Web.

Un fattore importante da considerare nella scelta dell'editor è la facilità di salvare i file di testo standard con l'estensione del nome file .html. Nel caso di Windows, qualsiasi programma che permetta non solo di salvare i file come testo per impostazione predefinita, ma che permetta anche di impostare l'estensione come .htm o .html, evita molti problemi. Se viene utilizzato Microsoft Word, per esempio, il programma cerca di salvare i file come file di Word binari, che nessun browser Web è in grado di caricare. Per salvare il file come testo o con l'estensione .html è necessario operare nella finestra di dialogo Salva con nome: è una vera seccatura.

Non c'è niente di male nell'utilizzare editor di testo essenziali. In Windows questi comprendono il programma WordPad o un prodotto più funzionale come l'editor shareware chiamato TextPad. Per Mac OS X va bene TextEdit. Uno dei preferiti tra gli autori HTML e i programmatori che utilizzano Mac è BBEdit (*Bare Bones Software*), che contiene numerosi ausili per i programmatori, come i numeri di riga facoltativi (utili quando viene eseguito il debug del codice JavaScript).

## Scelta di un browser

L'altro componente necessario per imparare JavaScript è il browser. Non è necessario essere connessi a Internet per provare gli script nel browser: è possibile eseguire tutti i test offline. Questo significa che è possibile imparare JavaScript e creare pagine Web interessanti e contenenti script anche con un computer portatile su una barca in mezzo all'oceano.

La marca e la versione del browser sono una scelta personale. Visto che i capitoli del tutorial in questo libro insegnano la sintassi del DOM di W3C, è opportuno procurarsi un browser recente. Uno qualsiasi dei seguenti potrà accompagnare correttamente i lettori nel tutorial: Internet Explorer 5 o successivi (per Windows o Macintosh), qualsiasi browser basato su Mozilla (compresi Netscape 7 e successivi), Apple Safari.

**NOTA**    *Molti listati di esempio nelle Parti III e IV del libro dimostrano funzioni del linguaggio o del modello di oggetti del documento (DOM) che funzionano solo con particolari browser o versioni. Conviene controllare sempre l'elenco di compatibilità per la funzione del linguaggio o del DOM per essere certi di utilizzare il browser corretto per il caricamento della pagina.*

## 3.2 Impostazione dell'ambiente di authoring

Per semplificare il lavoro di verifica degli script, bisogna disporre di sufficiente memoria libera nel computer per poter eseguire contemporaneamente l'editor di testo e il browser. È necessario essere in grado di passare velocemente dall'editor al browser durante le prove e la correzione degli errori presenti nel codice. Il flusso di lavoro tipico implica i seguenti passaggi:

1. Inserire il codice HTML e di scripting nel documento di origine con l'editor di testi.
2. Salvare l'ultima versione sul disco.
3. Passare al browser.
4. Eseguire una delle seguenti operazioni: se si tratta di un nuovo documento, aprire il file attraverso il comando **Apri** del browser; se il documento è già caricato, aggiornare il file nel browser.

I passaggi da 2 a 4 sono quelli che verranno seguiti più spesso. Questa sequenza di tre passaggi può essere definita sequenza di salvataggio-passaggio-aggiornamento: dovrà essere eseguita spesso durante lo scripting, e presto diverrà un'operazione naturale. La disposizione delle finestre delle applicazioni e l'esecuzione della sequenza di salvataggio-passaggio-aggiornamento cambiano secondo il sistema operativo.

### Windows

Non è necessario impostare le finestre dell'editor o del browser alle dimensioni massime (a schermo intero) per trarne vantaggio: in effetti è più facile lavorare se le dimensioni e la posizione di ciascuna finestra vengono regolate in modo che entrambe siano il più grandi possibile, pur permettendo di fare clic su una parte della finestra che non viene utilizzata. In alternativa, è possibile lasciare visibile la barra delle applicazioni, in modo da poter fare clic sul pulsante del programma desiderato per passare alla sua finestra (vedere la Figura 3.1). Un monitor che visualizza più di 800 x 600 pixel è sicuramente utile, perché offre più spazio per le finestre e la barra delle applicazioni.

Tuttavia, nella pratica, la scorciatoia da tastiera **ALT+TAB** per passare da un programma all'altro semplifica ancora di più l'esecuzione dei passaggi di salvataggio, passaggio e aggiornamento precedenti. Chi esegue Windows e utilizza un editor di testo compatibile con Windows (che molto probabilmente utilizza **CTRL+S** per il salvataggio dei file) può attuare la sequenza utilizzando solo la mano sinistra: **CTRL+S** (salva il file di origine); **ALT+TAB** (passa al browser); **CTRL+R** (ricarica il file di origine salvato). Finché il passaggio tra il browser e l'editor di testo viene eseguito con **ALT+TAB**, i programmi possono essere sempre raggiunti in modo molto semplice.

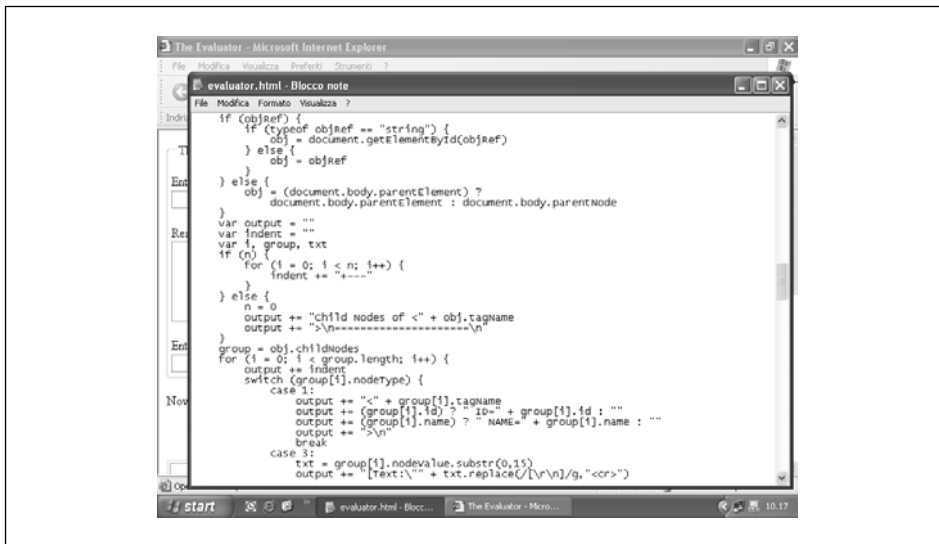


Figura 3.1 La disposizione delle finestre dell'editor e del browser in Windows XP.

## Mac OS X

In Mac OS X è possibile passare tra l'editor di testi e il browser utilizzando il Dock oppure, più comodamente, premendo **MELA+TAB**. Finché vengono utilizzate solo queste due applicazioni, l'altro programma può essere attivato premendo **MELA+TAB** (vedere la Figura 3.2).

Con questa impostazione, la sequenza di salvataggio, passaggio e aggiornamento è semplice:

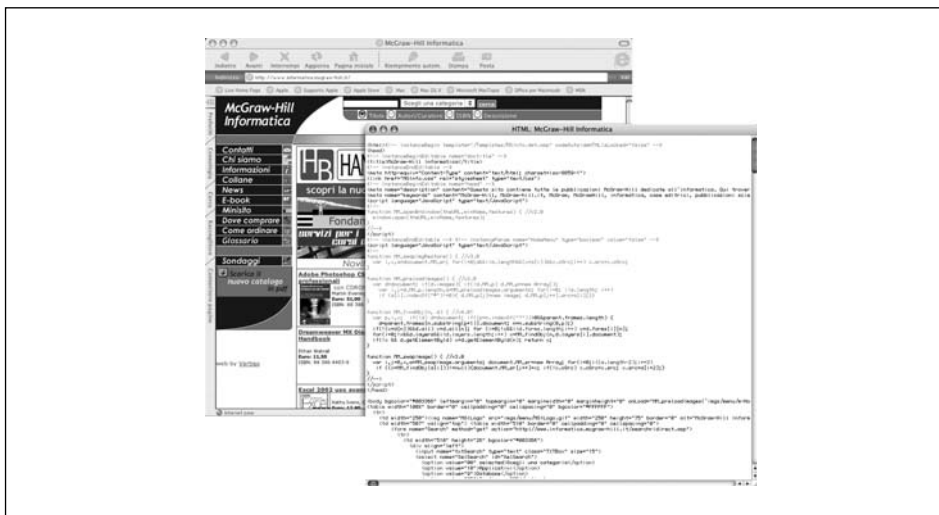


Figura 3.2 La disposizione delle finestre dell'editor e del browser sullo schermo di Macintosh.

1. Premere `MELA+S` (salva il file di origine).
2. Premere `MELA+TAB` (passa al browser).
3. Premere `MELA+R` (ricarica il file di origine salvato).

Per tornare a modificare il file di origine è sufficiente premere di nuovo `MELA+TAB`.

## Problemi relativi all'aggiornamento

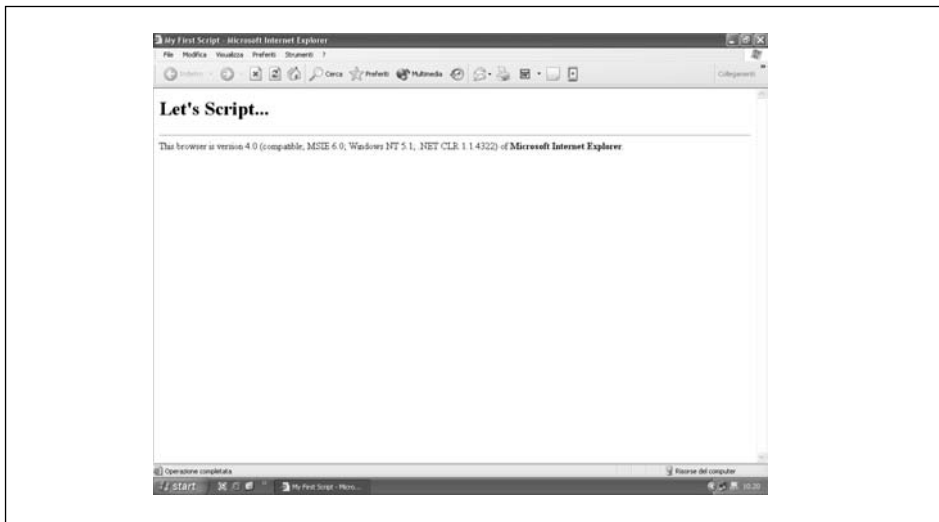
Nella maggior parte dei casi, per provare una versione rivista di uno script è sufficiente ricaricare semplicemente la pagina; a volte, però, quando si ricarica una pagina la cache del browser (con le sue impostazioni predefinite) può conservare parte degli attributi della pagina precedente, anche se è stato modificato il codice sorgente. Per eseguire un'operazione di ricarica più accurata, occorre tenere premuto `MAIUSC` mentre si fa clic sul pulsante **Aggiorna** del browser. In alternativa è possibile disattivare la cache del browser nell'area delle preferenze, ma questa impostazione può influire negativamente sulle prestazioni generali del browser durante le normali operazioni di esplorazione del Web.

### 3.3 Che cosa farà il primo script

Per amore della semplicità, il genere di script presentato nella prossima sezione viene eseguito automaticamente quando il browser carica la pagina HTML. Sebbene tutto il lavoro di scripting e di esplorazione sia svolto offline, il comportamento della pagina è identico se il file di origine viene caricato su un server e qualcuno vi accede attraverso il Web.

La Figura 3.3 mostra la pagina come appare nel browser dopo il completamento (le parole cambiano leggermente se si esegue il browser con un sistema operativo diverso da Windows XP o se viene utilizzato un browser diverso da Internet Explorer). La parte della pagina definita in HTML normale non contiene altro che un titolo di livello `<h1>` con un filetto orizzontale sotto esso. Se qualcuno non utilizza un browser compatibile con JavaScript, vedrà solo il titolo e il filetto orizzontale (a meno che abbia un browser veramente datato, nel qual caso vedrà alcune parole dello script nella pagina).

Sotto il filetto lo script visualizza un testo normale che combina il testo statico con informazioni sul browser utilizzato per caricare il documento. Lo script scrive un flusso di informazioni HTML nel browser, compreso un tag che applica un foglio di stile per la visualizzazione di una parte delle informazioni in grassetto. Anche se sono due righe di codice a scrivere informazioni nella pagina, il risultato viene rappresentato su una riga unica, esattamente come quando si scrive in HTML tutto il testo.



**Figura 3.3** La pagina finita del primo script in JavaScript.

### 3.4 Inserimento del primo script

È arrivato il momento di cominciare a creare il primo script in JavaScript. Avviare l'editor di testi e il browser; se il browser chiede di collegarsi al provider di servizi Internet (ISP, *Internet Service Provider*) o comincia automaticamente la composizione del numero, annullare l'operazione. Se è attivo il pulsante Termina del browser, fare clic su esso per fermare qualsiasi ricerca possa essere in corso. Potrebbe apparire una finestra di dialogo o una pagina con un messaggio che indica che l'URL della pagina iniziale del browser (in genere la home page del produttore del browser, a meno che siano state modificate le impostazioni) non è disponibile. È tutto OK. Lo scopo è aprire il browser, ma non è necessario essere connessi al proprio ISP. È tutto a posto anche se la connessione a Internet avviene automaticamente attraverso una rete locale in ufficio o a scuola, oppure tramite un modem via cavo o DSL. Ad ogni modo, per il momento non è necessaria una connessione di rete. Seguire questi passaggi per inserire e visualizzare in anteprima il primo script in JavaScript:

1. Attivare l'editor di testi e creare un nuovo documento vuoto.
2. Digitare lo script nella finestra esattamente come mostrato nel Listato 3.1.

**Listato 3.1** Codice sorgente per script1.html

```
<html>
<head>
<title>My First Script</title>
<style type="text/css">
```

```

.highlight {font-weight: bold}
</style>
</head>

<body>
<h1>Let's Script...</h1>
<hr>
<script type="text/javascript">
<!-- nascosto ai vecchi browser
document.write("Questo browser e' la versione " + navigator.appVersion);
document.write(" di <span class='highlight'>" + navigator.appName + "</
span>.");
// fine script nascosto -->
</script>
</body>
</html>

```

3. Salvare il documento con il nome script1.html.
4. Passare al browser.
5. Selezionare Apri (o Apri file con alcuni browser) dal menu File e selezionare script1.html (in alcuni browser è necessario fare clic sul pulsante Sfoglia per aprire la finestra di dialogo File).

Se tutte le righe sono state digitate come indicato, il documento nella finestra del browser dovrebbe somigliare a quello mostrato nella Figura 3.3 (con lievi differenze relative al sistema operativo e alla versione del browser). Se il browser indica l'esistenza di un errore quando il documento viene caricato, per il momento non bisogna fare niente (è sufficiente fare clic sul pulsante OK se appare una finestra di dialogo con un errore di scripting). Per prima cosa è opportuno esaminare i dettagli dell'intero documento, in modo da capire che cosa fa lo script.

### 3.5 Analisi dello script

Non è necessario memorizzare tutti i comandi o la sintassi discussi in questa sezione; piuttosto, è consigliabile rilassarsi e osservare come le righe dello script diventano ciò che appare nel browser. Nel Listato 3.1, tutte le righe fino al tag `<script>` sono codice HTML standard con una regola CSS (*Cascading Style Sheets*) nella parte dell'intestazione.

#### Il tag `<script>`

Ogni qualvolta occorre inserire codice JavaScript in un documento HTML, è necessario racchiuderlo all'interno di una coppia di tag `<script>...</script>`. Questi tag comunicano al programma del browser di iniziare a interpretare

come script tutto il testo tra i tag. Poiché altri linguaggi di scripting (come VBScript di Microsoft) possono utilizzare questi tag per gli script, è necessario specificare il tipo di linguaggio in cui è scritto il codice contenuto tra i tag. Di conseguenza, quando il browser riceve il segnale che lo script è di tipo `text/javascript`, per gestire il codice utilizza il suo interprete JavaScript incorporato. Nella vita reale è possibile trovare dei paralleli per questa impostazione: con un interprete francese al proprio fianco, è necessario sapere che anche la persona con la quale si sta conversando conosce il francese. Quando si incontra una persona proveniente dalla Russia, l'interprete francese non può essere di aiuto. Analogamente, se il browser ha al suo interno solo un interprete JavaScript, non è in grado di comprendere il codice scritto in VBScript.

Questo è un ottimo momento per introdurre un aspetto di JavaScript che sarà importante ogni qualvolta verrà utilizzato questo linguaggio: JavaScript fa distinzione tra maiuscole e minuscole. È quindi necessario inserire qualsiasi elemento dello script che utilizza una parola JavaScript con le lettere maiuscole e minuscole corrette. I tag HTML (compreso il tag `<script>`) possono essere maiuscoli o minuscoli, come si desidera, ma tutto quanto appartiene a JavaScript è sensibile alle maiuscole<sup>1</sup>. Quando una riga di codice JavaScript non funziona, in primo luogo è opportuno controllare maiuscole e minuscole. È necessario confrontare sempre il codice digitato con i listati stampati in questo libro e con le varie voci del vocabolario discusse al suo interno.

## Uno script per tutti i browser

La riga dopo il tag `<script>` nel Listato 3.1 sembra l'inizio di un tag di commento HTML. In effetti lo è, ma l'interprete JavaScript tratta i tag di commento in modo particolare. Anche se JavaScript ignora una riga che comincia con un tag iniziale di commento HTML, tratta la riga successiva come una riga di script a tutti gli effetti. In altre parole, lo strumento di gestione degli script all'interno del browser inizia ad interpretare la riga successiva a un tag iniziale di commento. Se si desidera inserire un commento all'interno del codice JavaScript, tale commento deve iniziare con una barra doppia (`//`). Questo commento può trovarsi verso la fine di una riga (dopo un'istruzione JavaScript che deve essere interpretata dal browser) o su una riga a sé stante. In effetti, la seconda situazione si verifica verso la fine dello script. La riga di commento inizia con due barre.

È opportuno tornare indietro un attimo e osservare che l'intero script (compresi i commenti) è contenuto all'interno di un tag di commento HTML standard (`<!--commento-->`). Il valore di tutto ciò non è chiaro fino a quando non si vede cosa accade al documento HTML contenente script in un browser non compatibile con JavaScript. Un browser del genere passa oltre il tag

<sup>1</sup> Lo stile XHTML, se si vogliono seguire queste convenzioni, richiede che i nomi dei tag e degli attributi siano in minuscolo. Questo è lo stile osservato in tutto il libro.

`<script>`, considerandolo come un tag incomprensibile, ma tratta una riga dello script come testo normale da visualizzare nella pagina. Se le righe dello script vengono racchiuse tra i tag di commento HTML, la maggior parte dei vecchi browser non visualizza le righe dello script.

È bene ricordare, comunque, che alcuni utenti non hanno accesso a browser moderni o grafici (utilizzano il software di lettura del Web UNIX orientato ai testi Lynx o browser leggeri sui computer palmari). Racchiudendo le righe dello script all'interno dei commenti, le pagine Web non hanno un aspetto completamente sbagliato nei browser relativamente moderni ma non compatibili con JavaScript.

**NOTA** *Occorre notare che le righe di commento che proteggono i vecchi browser dagli script vanno all'interno dei tag `<script>...</script>`. Non è possibile inserire tali righe di commento sopra il tag `<script>` o sotto il tag `</script>` e aspettarsi che funzionino.*

In questo libro viene affrontata un'altra questione riguardante le righe di commento che nascondono lo script. Per risparmiare spazio sulla pagina, la maggior parte degli esempi non contiene righe di commento. Nelle pagine prodotte per il pubblico, però, è necessario inserire sempre le righe degli script all'interno dei commenti.

## Visualizzazione di testo

Entrambe le righe dello script del Listato 3.1 utilizzano una delle possibili azioni che uno script può chiedere a un documento di eseguire (`document.write()`, che chiede di visualizzare testo nel documento corrente). Ulteriori informazioni sull'oggetto `document` sono disponibili nel Capitolo 18.

Quando si chiede a un oggetto (in questo caso un documento) di eseguire un'operazione, il nome dell'operazione è sempre seguito da una coppia di parentesi. In alcuni casi (per esempio l'operazione `write()`), JavaScript deve sapere su quale informazione deve agire: tale informazione (chiamata parametro) deve essere all'interno delle parentesi dopo il nome dell'operazione. Di conseguenza, per scrivere il nome del primo presidente degli Stati Uniti in un documento, il comando da utilizzare è il seguente:

```
document.write("George Washington");
```

La riga di testo scritta dallo script inizia con un testo statico ("Questo browser e' la versione") e aggiunge ad esso un testo calcolato (la versione del browser). La scrittura continua con altro testo statico che contiene un tag HTML (" di `<span class='highlight'>`"), altro testo calcolato (il nome dell'applicazione browser), un tag di chiusura HTML e il punto della frase ("`</span>.`"). JavaScript utilizza il segno più (+) per unire (*concatenare*) i componenti di testo in una singola stringa di caratteri di testo da scrivere nel

documento. Né JavaScript né il simbolo + conoscono parole e spazi, quindi lo script è responsabile dell'inserimento degli spazi necessari, passati come parte dei parametri. Occorre notare, quindi, che esiste uno spazio dopo la parola "versione" nel parametro della prima istruzione `document.write()` e ulteriori spazi prima e dopo la parola "di" nel parametro della seconda istruzione `document.write()`.

Per recuperare le informazioni sul nome e sulla versione del browser per i parametri, occorre chiedere a JavaScript di estrarre le proprietà corrispondenti dall'oggetto `navigator`. È possibile estrarre una proprietà aggiungendo il nome della proprietà al nome dell'oggetto (in questo caso `navigator`) e separando i due nomi con un punto. Chi volesse trovare un equivalente mentale per la lettura di questo schema, potrebbe iniziare da destra e definire l'elemento a destra come una proprietà del lato sinistro: la proprietà `appVersion` dell'oggetto `navigator`. Questa sintassi a punti assomiglia molto all'operazione `document.write()`, ma il nome di una proprietà non è seguito da parentesi. In ogni caso, il riferimento alla proprietà nello script comunica all'interprete JavaScript di inserire il valore di tale proprietà nel punto in cui viene eseguita la chiamata. JavaScript sostituisce l'informazione interna riguardante il browser come parte della stringa di testo che viene scritta nel documento.

Per finire, è possibile notare i caratteri di punto e virgola alla fine di ogni istruzione JavaScript. Questi punti e virgola sono completamente facoltativi; non vi sono problemi se vengono tralasciati. Osservando altri linguaggi di programmazione, come Java o C++, per esempio, è possibile scoprire che i punti e virgola sono necessari. I listati di questo libro utilizzano i punti e virgola.

### 3.6 Un po' di divertimento

Quando si incontra un errore nel primo tentativo di caricare questo documento nel browser, bisogna tornare all'editor di testi e confrontare le righe della sezione dello script con il Listato 3.1, osservando attentamente ciascuna riga alla luce delle spiegazioni. Vi potrebbe essere un carattere fuori posto, una lettera minuscola che dovrebbe essere maiuscola, o un carattere virgoletta o parentesi mancante. Occorre quindi eseguire le correzioni necessarie, tornare al browser e fare clic sul pulsante **Aggiorna**.

Per vedere l'aspetto dinamico dello script di `script1.html`, è necessario tornare all'editor di testi e sostituire la parola "browser" con "software client". Salvare, tornare al browser e ricaricare la pagina per vedere come lo script ha cambiato il testo nel documento. È possibile sostituire il testo tra virgolette con altro testo nell'istruzione `document.write()`, oppure aggiungere altro testo con ulteriori istruzioni `document.write()`. I parametri di `document.write()` sono testo HTML, quindi è anche possibile scrivere "`<br>`" per inserire un'interruzione di riga. È importante assicurarsi sempre di salvare, passare al browser e ricaricare la pagina per vedere i risultati del proprio lavoro.