

# HTML 5 e CSS 3



**Gabriele Gigliotti**

**SCOPRIRE**  
il linguaggio  
di markup che sta  
rivoluzionando il Web

**MIGLIORARE**  
l'esperienza utente  
con interfacce dinamiche  
e contenuti più ricchi

“Il Web non collega solo le macchine,  
collega le persone.”  
– Tim Berners-Lee

**APGEO**

# HTML 5 e CSS 3



Gabriele Gigliotti

## SCOPRIRE

il linguaggio  
di markup che sta  
rivoluzionando il Web

## MIGLIORARE

l'esperienza utente  
con interfacce dinamiche  
e contenuti più ricchi

“Il Web non collega solo le macchine,  
collega le persone.”  
– Tim Berners-Lee

**APGEO**

*Gabriele Gigliotti*

**APOGEO**

© Apogeo s.r.l. - socio unico Giangiacomo Feltrinelli Editore s.r.l.

ISBN edizione cartacea: 978-88-50-33011-9



# Prefazione

HTML e CSS. Due acronimi che racchiudono il DNA di un mondo, il Web.

Non si esagera definendoli le lingue franche del Web.

Senza di loro comunicare in Rete non sarebbe l'esperienza fantastica e variegata che oggi è. Niente siti, forum, blog, social network. Niente Google, Facebook, Twitter. Tutto sarebbe meno facile e più simile alla colata di codice dei terminali nella trilogia Matrix.

HTML e CSS sono belli perché rendono il Web bello e usabile. Ma sono anche magici perché facili metafore (ok, sarebbe meglio dire interfacce) tra chi scrive e chi legge.

Inoltre sono storici perché entrambi divennero raccomandazioni ufficiali nelle versioni ancora oggi utilizzate alla fine degli anni Novanta dello scorso secolo. Da allora abbiamo a disposizione HTML 4.01 e CSS 2.

Fino a oggi.

Perché HTML e CSS stanno di nuovo evolvendo e qua e là nel Web spuntano i segni di quanto HTML 5 e CSS 3 portano con sé.

Per chi fa Web il cambiamento è quasi copernicano, senza dubbio stupefacente. Mentre chi il Web lo usa solo, beh, si prepari pure a stupirsi.

HTML 5 e CSS 3: la magia si rinnova e in questo manuale Gabriele ci racconta i trucchi, che poi trucchi non sono, ma semplici elementi, attributi, selettori e proprietà che spalancano possibilità fino a ora solo immaginate o perseguitate a fatica attraverso codici complessi e hack.

E, cosa più importante, trucchi che si possono usare. Subito.

Fabio Brivio  
Editor

[fabio@apogeonline.com](mailto:fabio@apogeonline.com)

# Ringraziamenti

A Erica, instancabile corretrice di bozze.

A Giordano, impareggiabile occultatore di bozze.

A Fabio Brivio che ha creduto sin dal principio in questo progetto, mi ha incoraggiato e mi ha regalato la sua prefazione.

A Federica Dardi, per l'entusiasmo e la pazienza con cui ha curato la revisione del manuale.

A Alberto Kratter Thaler che ha intuito il potenziale del manuale HTML 4.01 e mi ha dato la possibilità di pubblicarlo.

A Lorenzo che mi ha guidato nei meandri delle lingue ideografiche.

A Giuseppe che mi ha insegnato a usare la rete e mi ha prestato il primo libro di HTML. Guarda cosa hai combinato!

A Voi tutti grazie. Senza di voi questo manuale non esisterebbe.

# Capitolo 1

## HTML: How Tough and Marvellous this Language is! (version n. 5)

Il titolo di questo capitolo è un tributo al linguaggio di marcatura che ha tramutato in realtà quell'ipertesto solo vagheggiato da Vannevar Bush nel 1945. HTML, acronimo che in realtà significa HyperText Markup Language (linguaggio di marcatura per ipertesti), sta dimostrando di sapersi rinnovare e portare il Web oltre il 2.0, verso la nuova frontiera del Web semantico.

La storia di HTML è così affascinante e al tempo stesso complessa da meritare un testo a parte. Il manuale che avete tra le mani, invece, ha un taglio decisamente pratico. Mi limito solo a osservare che, a causa dei non sempre felici rapporti tra il WHATWG e il W3C (i due enti coinvolti nella definizione della specifica), del fatto che le aziende coinvolte nello sviluppo dei principali browser hanno interessi divergenti, dell'eredità lasciata dalle centinaia di milioni di pagine web oggi esistenti cui HTML5 garantisce compatibilità, il linguaggio di marcatura risponde adottando scelte che possono sembrare discutibili e forse, in alcuni casi, ne minano la consistenza. Sono tuttavia frutto di un sano pragmatismo, senza il quale avrebbero potuto iniziare interminabili guerre di religione.

Pragmatismo: questa è una delle parole chiave che hanno guidato lo sviluppo della nuova versione del linguaggio. Ciò ha significato prendere atto del potere discrezionale di cui godono i principali attori sul mercato dei browser nel decidere che cosa implementare e che cosa eliminare. Il tutto, partendo dalla constatazione che "non ha senso tradurre in specifica qualcosa che non può essere utilizzato perché una parte significativa dei visitatori non ne disporrà mai. Meglio dedicarsi a tradurre in specifica cose che tutti implementeranno effettivamente" e sempre allo scopo di "rendere la specifica rispondente alla realtà" (<http://lists.whatwg.org/htdig.cgi/whatwg-whatwg.org/2010-June/026897.html>).

Un secondo principio cardine è stato il supporto diretto di tutte quelle soluzioni in cui JavaScript ha dimostrato di saper svolgere con efficacia il proprio compito.

Esempi di questa scelta sono evidenti nelle nuove funzionalità aggiunte ai form; per esempio, gli attributi `placeholder`, `autofocus` e `required` nascono proprio dalla consapevolezza che il diffuso impiego di JavaScript rivelava una lacuna delle precedenti versioni di HTML. Su questo tema si dirà di più nel Capitolo 4.

Un terzo pilastro della specifica è quello della semantica, tema ricorrente in diversi capitoli di questo manuale. HTML5 pone le basi per una migliore definizione dei vari componenti di una pagina web. I tag e gli attributi HTML vengono impiegati sempre più

per fornire informazioni aggiuntive sui numerosi dati che popolano ogni pagina web di cui curiamo la pubblicazione. Una maggiore sensibilità verso questo aspetto significa aprire le porte a motori di ricerca più efficienti: si pensi al modo in cui già da più di un anno Google usa RDFa, microformat e microdata (formati di definizione e marcatura che definiscono uno standard per lo scambio di dati, in modo automatico, tra macchine) per offrire un set arricchito di informazioni partendo dai dati strutturati contenuti nelle pagine web create da ciascuno di noi. La semantica è anche la via attraverso la quale “liberare i dati” rendendone possibile la fruizione non solo a individui, ma anche direttamente ad altre macchine; in questo modo i dati possono essere rielaborati per assolvere funzioni diverse da quelle per cui originariamente erano stati resi disponibili. Infine, una più accurata descrizione del significato di un documento significa anche maggiore accessibilità per mezzo di tecnologie di supporto che, sfruttando una più approfondita conoscenza delle pagine web, possono svolgere meglio la loro funzione di supporto alla navigazione.

Uno degli scopi di questo manuale è di documentare che cosa si può fare già oggi con HTML5.

Per guidarvi lungo questo percorso troverete due utili strumenti, illustrati di seguito.

- **Tabelle di compatibilità dei browser.** Sono necessarie per capire con un colpo d’occhio quanto sia diffuso il supporto di una data funzionalità. Nel testo troverete sintetiche tabelle di compatibilità utili per farsi un’idea del supporto delle singole funzionalità man mano che queste sono introdotte. L’insieme di elementi e attributi implementati aumenta di pari passo con il susseguirsi dei rilasci di nuove versioni dei browser. Per questo motivo, considerate le tabelle solo come un punto di partenza, che non può sostituire test eseguiti direttamente sulle principali combinazioni di browser e sistemi operativi.
- **Strategie di gestione delle incompatibilità.** In una sola parola: retro-compatibilità. Bisogna prendersi cura di quegli utenti che non possono o non vogliono cambiare browser. In questi casi ricorreremo a Modernizr, una libreria JavaScript che ci aiuta a identificare il supporto delle funzionalità introdotte con HTML5 e CSS3.

## Una panoramica sulle differenti piattaforme web

Per testare accuratamente il comportamento e la visualizzazione delle pagine web sviluppate anche sulle più improbabili combinazioni di browser e sistemi operativi, il modo migliore consiste nel rivolgersi a uno dei diversi servizi promossi in Rete. Ne esistono sia gratuiti, come [browsershots.org](http://browsershots.org), sia a pagamento, come [browsercam.com](http://browsercam.com) e [CrossBrowserTesting.com](http://CrossBrowserTesting.com). Questi ultimi offrono una modalità “dal vivo”, solitamente differenziandosi in questo dai servizi gratuiti. Con tale opzione si ottiene l’accesso remoto a un computer con la combinazione browser/sistema operativo desiderata, così da testare in concreto come risponde l’interfaccia utente in un contesto che altrimenti potrebbe essere difficile replicare. In aggiunta a ciò, si trova il classico servizio che “cattura” l’immagine del browser nell’atto di visualizzare il documento web. I prezzi variano tra i 20 e i 40 dollari al mese. È probabile che l’investimento non sia giustificabile per lo sviluppo di un sito hobbyistico, dove le risorse economiche tendono a zero; in casi come questi, il suggerimento è di sollecitare il prezioso riscontro degli utenti. Si potranno così ottenere informazioni per migliorare la fruibilità dei propri contenuti e al contempo fidelizzare i visitatori; è quel che si dice “saper fare di necessità virtù”.

Qualunque sia il browser che state utilizzando, è probabile abbiate già strumenti di ausilio allo sviluppo. È quanto accade, per esempio, in Chrome e in Opera. Per Firefox esistono diversi componenti aggiuntivi, tutti molto validi, che offrono supporto in quest'area. Firebug è quello che più di altri dovrebbe meritare la vostra attenzione. Per Internet Explorer fino alla versione 7 è necessario scaricare un componente aggiuntivo, mentre dalla versione 8 in avanti trovate quanto vi serve già integrato nel browser. Il motivo per cui vi invito caldamente a utilizzare questi strumenti è che essi aumentano la produttività consentendovi di identificare e risolvere bachi in tempi ridotti, e rappresentano un'insostituibile fonte di apprendimento su come operano le applicazioni web. Dedicare del tempo a familiarizzare con queste funzionalità è un investimento che si ripaga in pochissimo tempo.

## Modernizr vs soluzioni fai da te

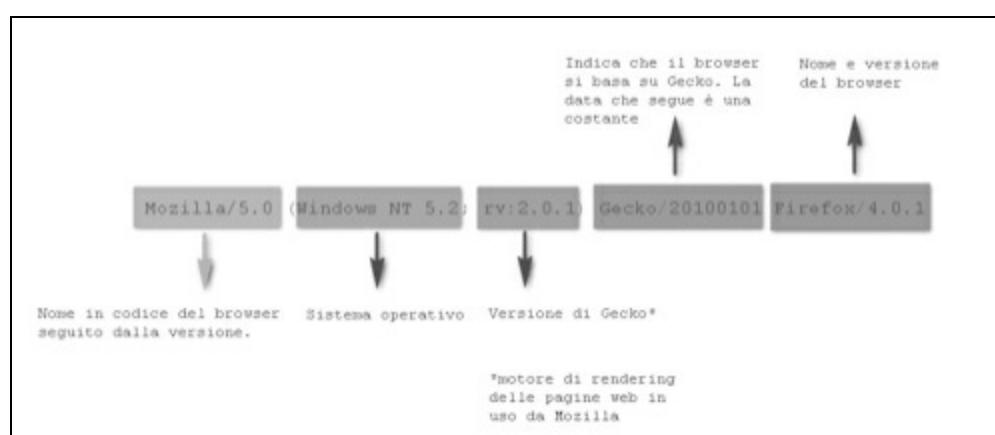
Capire se e quando è il caso di fornire una soluzione alternativa a quanto di nuovo offerto da HTML5 è il primo, ovvio, passo per l'adozione di una strategia di retrocompatibilità. A grandi linee, possiamo affermare che esistono due diverse strade per raggiungere l'obiettivo: una è l'identificazione del browser (tecnica diffusissima, sebbene sconsigliata in questo contesto), l'altra è il test del supporto dell'implementazione condotto sulla singola funzionalità (la cosiddetta feature detection).

## Come ti scopro il browser!

Quale browser sta visitando la nostra home page? Qual è la versione? E ancora, su quale sistema operativo è installato? Si tratta di un sistema a 32 o 64 bit? Le risposte a queste domande sono contenute come per magia in un'unica stringa di testo che prende il nome di stringa User Agent.

### **LISTATO 1.1** Un esempio di stringa User Agent

Mozilla/5.0 (Windows NT 5.2; rv:2.0.1) Gecko/20100101 Firefox/4.0.1



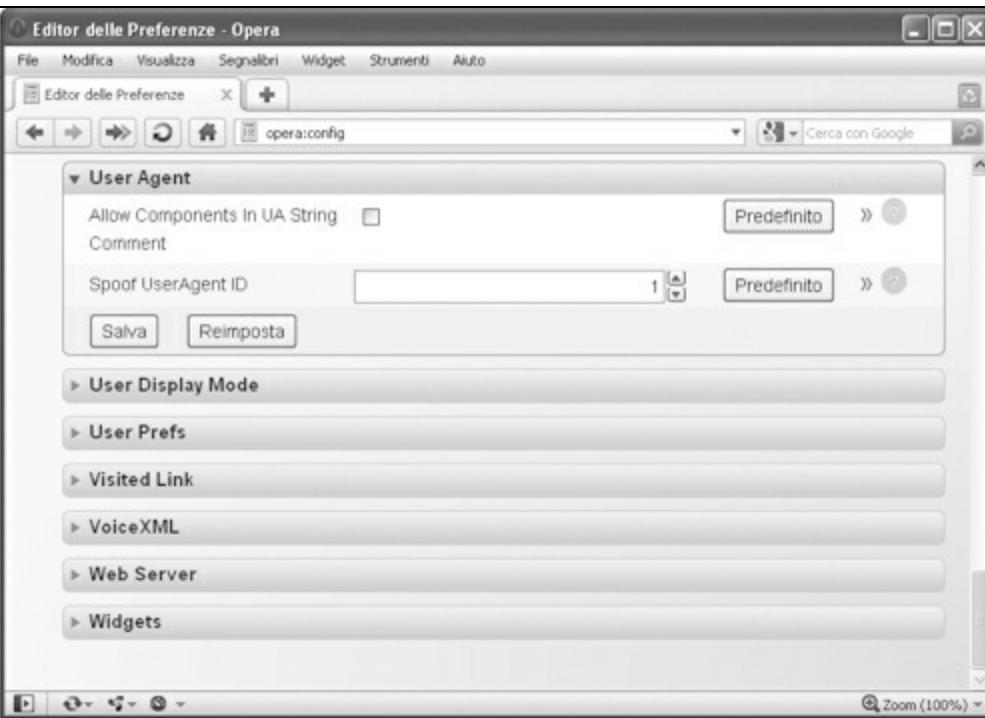
Per una panoramica degli elementi che la compongono si può osservare la Figura 1.1. L'analisi delle stringhe User Agent al fine di stabilire se un browser supporta o meno una data funzionalità HTML5 non è la soluzione migliore, per diversi motivi:

- perché l'identificazione del browser sia solida occorre sviluppare script complessi;
- è necessario mettere in preventivo una frequente attività di manutenzione a causa delle modifiche periodicamente apportate a queste stringhe, sia per quanto riguarda il numero di informazioni proposte, sia per quanto riguarda la loro disposizione all'interno delle stringhe stesse (per quanto ciò non accada spesso, ogni volta che si verifica viene minata alla base tutta la tecnica di identificazione);
- come se non bastasse, alcuni browser hanno adottato la politica di "assumere le sembianze" di altri browser mimandone le relative stringhe User Agent (Figure 1.2 e 1.3).

Le possibilità di mascherare il browser sono una risposta alla domanda degli utenti di poter scavalcare le barriere poste dagli autori alla segregazione dei contenuti sulla base di tipo e versione del browser.

È sempre rischioso proporre funzionalità e contenuti diversi in base alla "marca e modello" del software, non solo per l'aggravio dei costi di manutenzione di una strategia non scalabile (è altamente probabile che al rilascio di ogni nuova versione di un browser si debba operare la revisione del codice), ma anche perché, pur riuscendo a concepire la più raffinata strategia di identificazione, non si avrà mai la certezza che la funzionalità di cui intendiamo testare il supporto sia effettivamente stata implementata o meno. Il solo ragionevole campo di applicazione del browser, oggi, resta quello dell'identificazione fine a se stessa per meri scopi statistici.

Ce n'è quanto basta per poter affermare che ogni tecnica di identificazione del browser si presta a un certo grado di approssimazione che possiamo risparmiarci, se decidiamo di optare per una più mirata tecnica di identificazione della singola funzionalità.



**FIGURA 1.2** Scegliendo un valore da 2 a 5 nell'editor delle preferenze di Opera si ottengono altrettante stringhe User Agent.



**FIGURA 1.3** È possibile installare in Firefox un add-on che gli permette di assumere le sembianze di altri browser.

## Più browser sul mercato, più screenshot da browser diversi

Pur senza entrare nel dettaglio dell'evoluzione delle quote di mercato dei browser degli ultimi anni, oggi ci troviamo davanti a uno scenario più complesso di quello esistente solo pochi anni fa, quando Internet Explorer deteneva percentuali bulgare di base installata. Dalla rilevazione annuale condotta nel mese di ottobre 2010 da StatCounter, un servizio gratuito di statistiche di accesso a siti web, emerge che, per la prima volta, i browser della Microsoft (senza distinzione di versione) sono scesi nel loro complesso sotto la soglia del 50%, attestandosi al 49,87%. Si tratta pur sempre di una quota rilevante del totale dei browser, ma lontana anni luce dai fasti della fine anni Novanta e inizio 2000; le cifre sono ancora più negative per il colosso di Redmond se si guarda solo all'Europa, dove la percentuale di diffusione di Internet Explorer scende fino al 40,26%. Sono diventati attori di primo piano Firefox (31,5%) e Google Chrome (11,54%); Safari e Opera, seppure con percentuali inferiori, sono gli altri tasselli di un mosaico del quale è

## Identificazione diretta della funzionalità

Più efficace dell'identificazione del browser risulta, come detto, il test diretto del supporto di una funzionalità mediante gli oggetti, le proprietà e i valori che ne sono espressione. In parole povere, ci si concentra su che cosa un browser sa realmente fare, disinteressandosi del resto. Seguendo questa filosofia, Modernizr, la libreria JavaScript, esegue rapidamente una serie di test che mirano a stabilire se una data funzionalità sia supportata o meno. I risultati di tali test sono resi disponibili attraverso un elenco di proprietà booleane (ossia che possono assumere solo due valori: "vero" o "falso") di un oggetto chiamato appunto `Modernizr`.

Per esempio, se si volesse testare il supporto per l'elemento `<video>`, si potrebbe scrivere:

### **LISTATO 1.2** Test di supporto dell'elemento `<video>` via Modernizr

```
<script>
if (Modernizr.video) {
  /* video supportato */
} else {
  /* mancato supporto: applicazione di strategie alternative */
</script>
```

La proprietà booleana `video` dell'oggetto `Modernizr` restituirà `true` se l'elemento `<video>` è supportato dal browser, `false` in caso contrario. Nel Capitolo 4, relativo alle potenzialità multimediali di HTML5, vedremo come utilizzare questa libreria anche per altre necessità.

Va da sé che questo approccio semplifica molto il lavoro dell'autore di pagine web, il quale dovrà, in questo caso, limitarsi a controllare gli aggiornamenti di una sola libreria JavaScript, ed eventualmente intervenire con soluzioni ad hoc solo in caso di bachi.

Nel resto del presente manuale si utilizzerà questa libreria piuttosto che soluzioni di scripting "fai da te". A ogni modo, se nell'esempio sopra riportato si fosse voluto fare a meno di librerie esterne, si sarebbe potuto scrivere:

### **LISTATO 1.3** Script di identificazione del supporto all'elemento `<video>` (versione pedante)

```
<script>
function isTagVideoSupported() {
  var video;
  var bool;
```

```
video = document.createElement("video");
bool = new Boolean(video.canPlayType);

return bool;
}

</script>
```

oppure si sarebbe potuto optare per il meno pedante e più sintetico:

#### **LISTATO 1.4** Script di identificazione del supporto all'elemento `<video>` (versione "fate largo vado di fretta")

```
<script>
function isTagVideoSupported() {
    return !!document.createElement("video").canPlayType;
}
</script>
```

Indipendentemente dallo stile adottato, in questo caso notiamo come si tenda a verificare il supporto dell'elemento `video` leggendo il valore restituito da un suo metodo, `canPlayType`.

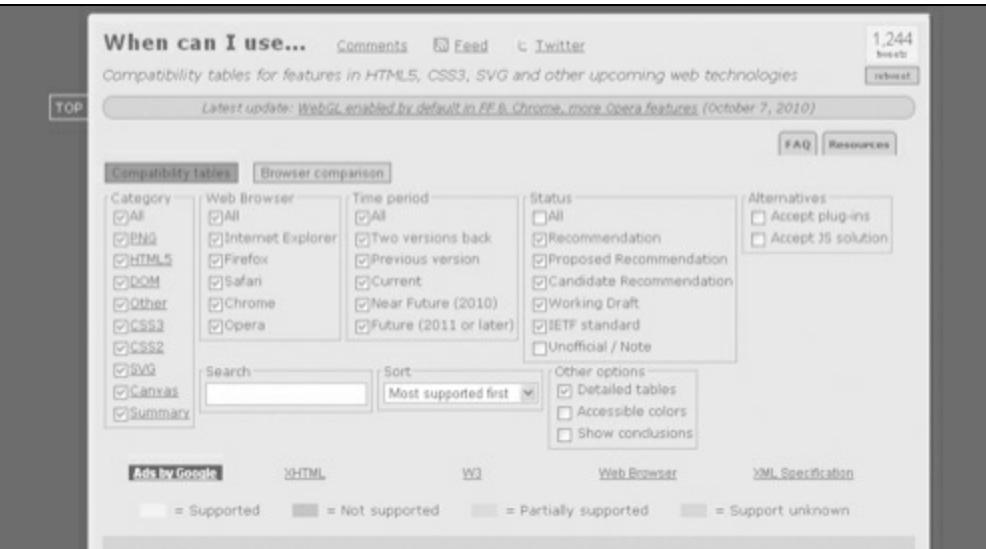
Questa è una delle quattro opzioni attraverso le quali valutare il supporto di una data funzionalità. Altre strategie verranno illustrate nel prosieguo del manuale.

Se desiderate stare alla larga da questa complessità, lavorando a un livello di astrazione più alto, e sviluppando al contempo un codice più sintetico e leggibile, oggi Modernizr (<http://www.modernizr.com/>) è la soluzione migliore.

## Farsi un'idea delle funzionalità supportate

Prima ancora di immergersi nel lavoro di implementazione, almeno in questa fase in cui, seppure a fronte di un crescente supporto di HTML5, non mancano alcune voci critiche che invitano a raffreddare facili entusiasmi su questa tecnologia, può essere d'aiuto sondare il supporto di una data funzionalità consultando alcuni pratici servizi come *When can I use* (Figura 1.4).

Una suggestiva rappresentazione dei dati impiegati dal progetto *When can I use* è riprodotta sul sito *HTML5 Readiness* che, facendo ricorso a una infografica dinamica, permette di cogliere immediatamente il grado di supporto delle principali funzionalità introdotte con l'ultima versione del linguaggio (Figura 1.5).



**FIGURA 1.4** Guida pratica di compatibilità a HTML5, CSS3 e altre tecnologie legate al Web.



**FIGURA 1.5** Un'accattivante rappresentazione grafica del livello di supporto rappresentato in base ai browser più diffusi e all'anno di riferimento.

# Librerie JavaScript

In questa fase di transizione che stiamo vivendo, in cui sezioni della specifica HTML5 sono utilizzabili subito mentre altre presentano un supporto ancora limitato, è naturale che si diffondano una serie di utili librerie JavaScript, che operano in tre diverse aree.

1. **Testare il supporto delle nuove funzionalità:** è il caso della libreria Modernizr, che non interviene su ciò che un dato browser è in grado di fare, limitandosi a verificarne il riconoscimento di specifiche funzioni.
  2. **Sopperire alla mancata implementazione di date funzionalità, "abilitando" il browser all'uso di tecnologie per cui altrimenti non sarebbe pronto:** rientra in questa categoria la libreria excanvas, che apre l'elemento `<canvas>` (cui si farà cenno nel Capitolo 6) al mondo dei browser di casa Microsoft, almeno nelle versioni precedenti alla 9.

3. Sperimentazione e applicazioni artistiche: sono queste le librerie più fantasiose che portano agli estremi una data tecnologia per enfatizzarne le potenzialità o solo per dimostrare come l'arte possa sposarsi anche con il Web. Un esempio in questo senso ci è offerto dalla libreria close-pixelate di David De Sandro.

La promessa di queste soluzioni sembra allettante: lasciare gli autori liberi di utilizzare subito il nuovo linguaggio, senza doversi preoccupare di sottoporre il codice a revisione quando i visitatori decideranno di aggiornare il proprio browser.

Proponiamo in Tabella 1.1 un elenco di alcune librerie che verranno discusse in questo manuale.

**TABELLA 1.1** Alcune librerie citate nel manuale

LIBRERIA	URL	DESCRIZIONE
modernizr	<a href="http://www.modernizr.com/">http://www.modernizr.com/</a>	Troveremo riferimenti a questa libreria di identificazione delle funzionalità supportate in diversi capitoli di questo manuale.
popcorn	<a href="http://mozilla.github.com/popcorn-js/">http://mozilla.github.com/popcorn-js/</a>	Nel capitolo relativo agli elementi multimediali, questa libreria è utilizzata per aggiungere i sottotitoli a un filmato.
excanvas	<a href="http://code.google.com/p/explorercanvas/">http://code.google.com/p/explorercanvas/</a>	Realizzata da Google, porta l'elemento <code>&lt;canvas&gt;</code> nel mondo Microsoft, almeno per le versioni precedenti alla 9.

## Sicurezza e trattamento dei dati personali

La progressiva diffusione del linguaggio HTML5 e delle tecnologie a esso collegate, come la geolocalizzazione, il Web storage e la possibilità di fruire di alcune funzionalità anche quando non connessi, rendono più sensibile il tema della privacy. Consultare siti web diventerà un'esperienza più coinvolgente, ma il prezzo che potremmo essere chiamati a pagare è quello dell'esposizione di una mole di dati relativi alle nostre abitudini e ai nostri interessi nettamente superiore a quella che possiamo inavvertitamente lasciare sulla Rete oggi. Se i cookie, file di pochi kilobyte in cui possono essere salvate le credenziali di accesso al sito, i dati relativi all'ultimo accesso e poco altro, destano ancora timore ma sono ormai da tempo ampiamente monitorabili attraverso il pannello di controllo di ogni browser, che cosa dire dei cookie zombie?

### Evercookie: il cookie che non muore mai

Samy Kamkar, lo sviluppatore che ha ideato l'interfaccia di programmazione alla base di questi cookie con i superpoteri, ha voluto dimostrare come sia possibile creare un sistema di persistenza dei dati realmente efficace. I cookie generati in questo modo saranno salvati sulla macchina dell'utente in punti diversi utilizzando svariate tecniche. Se uno o più file dovessero essere rimossi, evercookie utilizzerebbe una delle altre copie ancora presenti per replicarsi nuovamente.

# Riferimenti alle risorse citate e altri link utili

- Il sito del Web Hypertext Application Technology Working Group (<http://www.whatwg.org>), l'ente che ha curato sin dai primi passi la nuova specifica HTML5.
- L'ultima versione della specifica mantenuta dal W3C è disponibile presso: <http://www.whatwg.org/specs/web-apps/current-work/multipage>
- Se avete intenzione di seguire da vicino le persone coinvolte nella definizione delle specifiche, potreste trovare utile consultare i log delle conversazioni avute sul canale irc # whatwg: <http://krijnhoetmer.nl/irc-logs>
- Il World Wide Web Consortium, l'altro ente che, con il WHATWG, è coinvolto nella definizione della specifica, dispone di un sito web consultabile a questo indirizzo: <http://www.w3.org>
- La versione della specifica ospitata sul sito del W3C: <http://www.w3.org/TR/html5>
- Il grafico elaborato da StatCounter che esprime le quote di mercato dei browser: <http://gs.statcounter.com>
- When can I use, è un servizio che documenta il grado di supporto raggiunto da una data funzionalità HTML5 e CSS3: <http://caniuse.com>
- Una rielaborazione grafica interattiva dei dati riportati sul sito When can I use è disponibile a questo indirizzo: <http://html5readiness.com>
- Modernizr, ora giunta alla versione 1.6, è la libreria che useremo in diversi capitoli del manuale per testare, in concreto, il supporto delle funzionalità introdotte: <http://www.modernizr.com>
- Firebug integra un set di strumenti utili per analizzare le pagine web; che si tratti di HTML, CSS o JavaScript, questo componente aggiuntivo del browser Firefox riesce a dare sempre un valido supporto: <http://getfirebug.com>

## Conclusioni

È un momento davvero eccitante per gli autori di pagine web. Molte altre tecnologie legate al Web stanno raggiungendo un grado di maturità sufficiente per renderne possibile l'implementazione (geolocalizzazione, data storage, applicazioni web utilizzabili off-line). E ancora, ricordate l'ultima volta che è stato introdotto un nuovo formato di immagini per il Web? Penso che l'ultimo sia stato PNG, introdotto verso la fine degli anni Novanta (del secolo scorso dunque!!). Il 30 settembre 2010 Google ha annunciato il formato WebP, più performante del JPEG. Mantenendo l'attenzione su HTML, si nota come in nessuna delle precedenti versioni del linguaggio siano state introdotte così tante e profonde innovazioni. Inoltre, il ciclo di vita di molti browser si è ridotto, e questo rende più rapido l'estendersi del supporto di nuove funzionalità da parte dei principali browser in

commercio.

## Cicli di rilascio

Google Chrome, uno dei browser che negli ultimi anni ha conosciuto i maggiori tassi di crescita, ha finora rilasciato versioni stabili del proprio browser ogni tre mesi e di recente ha annunciato di aver ulteriormente ridotto questo ciclo di vita a sole 6 settimane.

I successivi capitoli, a partire dal prossimo, dedicato alla semantica dei documenti web, daranno un'idea più precisa del perché di tanto entusiasmo attorno a questo linguaggio. HTML5 rende possibile lo sviluppo di applicazioni web in modo più semplice e robusto di quanto non fosse possibile concepire anche solo fino a poco tempo fa.

Se nei prossimi anni assisteremo a una massiccia diffusione di applicazioni basate sulla specifica HTML5 e delle tecnologie web a essa legate, il Web come lo conosciamo oggi ci sembrerà preistoria.

# Capitolo 2

## Semantica

La semantica dei nuovi tag, ossia il significato che questi elementi attribuiscono al contenuto che demarcano, è uno degli aspetti più interessanti di HTML5. I nuovi tag si spogliano infatti di ogni implicazione stilistica: quest'ultima è un compito che viene demandato ai fogli di stile. Secondo questa filosofia, si innesca in modo decisivo il processo della separazione tra contenuto e sua visualizzazione, processo che ha avuto inizio con l'introduzione del CSS nel lontano 1996.

Solo i meno giovani tra voi ricorderanno i terrificanti tag `<blink>` o `<marquee>` (il che in fondo è un bene!), tuttavia anche altri elementi come `<strike>` o `<u>`, che nelle precedenti versioni del linguaggio sottintendevano una precisa modalità di visualizzazione del testo (testo barrato nel primo caso e sottolineato nel secondo), sono da considerarsi come appartenenti a un modo vetusto di realizzare documenti HTML.

I tag creati per contrassegnare le aree logiche in cui viene suddivisa una pagina web nella fase di disegno prendono il nome di elementi di struttura.

Nelle prossime pagine si approfondiranno dapprima gli elementi `<header>`, `<footer>`, `<nav>`, `<section>`, `<article>`, per poi procedere con un'analisi dei nuovi tag che migliorano ulteriormente la semantica degli ipertesti: `<figure>`, `<figcaption>` sono solo alcuni di questi.

Esaminiamo, quindi, quegli elementi che continuano a esistere, seppure con un significato diverso rispetto a quello assunto nelle precedenti versioni.

Il capitolo si chiuderà con un elenco di elementi e attributi dichiarati obsoleti, di cui si scoraggia l'utilizzo.

**TABELLA 2.1** Supporto degli elementi di struttura

ELEMENTO SUPPORTATO	BROWSER
Elementi di struttura	Chrome 5+, Firefox 3.0+, IE9+, Opera 10.1, Safari 3.2

### Elementi di struttura

Per disegnare la "gabbia", ossia la struttura di una pagina web, si è inizialmente fatto ricorso alle tabelle per garantire un'adeguata disposizione dei suoi elementi.

Costringendo in righe e colonne le diverse sezioni di cui si compone un documento HTML è stato possibile ottenere un'impaginazione che, seppure con un certo grado di approssimazione, riusciva a ricalcare quella tipografica. Gli svantaggi di questo approccio sono stati principalmente tre:

- l'annidare tabelle all'interno di tabelle, poste a loro volta all'interno di altre tabelle, ha messo alla corda i motori di interpretazione delle informazioni dei browser, i cosiddetti motori di rendering, determinando un aumento dei tempi di completo scaricamento della pagina web;
- questa tecnica ci ha regalato pagine web da cardiopalma: tempi lunghi e costi alti per ogni modifica che avesse un impatto sul layout;
- i tag usati per marcire il contenuto si mescolano con quelli utilizzati per puro scopo di visualizzazione, con buona pace di chi vuole dare nuova vita ai dati di cui si compongono gli ipertestî per scopi più evoluti.

## Spaghetti e zuppe da indigestione

Spaghetti code e tag soup sono due espressioni anglosassoni. Con la prima si vuole indicare un codice che, con una certa dose di eufemismo, possiamo definire poco leggibile, inutilmente complicato. Con la seconda si fa riferimento più specificatamente a HTML per indicare documenti zeppi di errori: elementi annidati in modo errato, non chiusi quando dovrebbero esserlo o usati a sproposito al solo scopo di produrre un determinato layout; sfortunatamente la lista è lunga!

La graduale sostituzione delle tabelle con il tag `<div>`, una sorta di contenitore generico, presto utilizzato per identificare aree della struttura di cui si compone la pagina web, ha aiutato a realizzare documenti HTML meno complessi. Tuttavia, anche in questo caso il `<div>` è stato travolto da orde di "annidatori" che hanno ecceduto nell'inserire `<div>` all'interno di altri `<div>`, appesantendo oltre il dovuto la pagina. Ciò ha portato a ipertestî più complessi di quelli che si sarebbe potuto ottenere attraverso uno studio più attento degli elementi del documento web, ma soprattutto non ha spostato di molto i termini del problema posti da quella continua alternanza di tag di presentazione e tag di pura semantica, ossia di definizione del contenuto. È accaduto insomma che, per una corretta visualizzazione della pagina, si sia fatto un uso smodato di classi di stile per contrassegnare le aree principali del documento (per esempio, tag come `<div class="header">` per indicare l'intestazione del documento, piuttosto che `<div class="navbar">`, sono esempi, molto diffusi, di questa pratica) perpetuando quella mescolanza di tag di contenuto con tag, e relativi attributi, di presentazione.

È proprio sulla base di queste esperienze che HTML5 introduce una serie di nuovi elementi il cui scopo è di contrassegnare aree della pagina web in modo non neutro, a differenza di quanto accadeva per il tag `<div>`. Un accorto utilizzo di questi nuovi tag di struttura, unitamente ai nuovi selettori CSS3, abbatte drasticamente l'esigenza di marcatori di pura presentazione posti direttamente all'interno del documento e assicura una più alta definizione del significato dei contenuti: in altri termini, una migliore

semantica.

Questi elementi non forniscono altro valore aggiunto se non quello di una più precisa marcatura dei contenuti. Il vantaggio immediato è una maggiore affidabilità delle tecnologie assistive, che potranno identificare in modo netto le diverse aree di cui si compone un documento e, conseguentemente, guidare in modo più efficace l'utente. È ipotizzabile che in futuro possa esserci un impatto anche sull'algoritmo di indicizzazione delle pagine web e, di conseguenza, anche sui risultati. Google, Bing (Yahoo! ormai adotta le tecnologie di ricerca di quest'ultimo) e altri motori di ricerca sapranno rispondere meglio alle nostre richieste.

## Chi ben incomincia...

La prima riga di un documento HTML identifica il doctype. Si tratta di una dichiarazione del tipo di documento che ha il duplice scopo di predisporre la pagina web per la validazione e al contempo di attivare la modalità standard di interpretazione del documento da parte di alcuni browser.

La definizione del tipo di documento per HTML5 è sorprendentemente semplice:

### **LISTATO 2.1** Dichiarazione DOCTYPE per HTML5

```
<!DOCTYPE html>
```

Tutto qui! Dichiarazione breve e facile da ricordare. Se pensate stia esagerando con tanto entusiasmo, vi ricordo alcuni doctype in uso per le precedenti versioni del linguaggio:

### **LISTATO 2.2** DOCTYPE relativo a HTML 4.01 strict

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

### **LISTATO 2.3** DOCTYPE relativo a HTML 4.01 Transitional

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01Transitional//EN"  
"http://www.w3.org/TR/HTML4.01/loose.dtd">
```

Si può dire, senza tema di smentita, che finora la dichiarazione doctype non sia stata certo semplice da memorizzare, nemmeno per i professionisti.

Nell'intestazione del documento, ossia tra i tag `<head>...</head>`, possono essere definiti i metadati del documento stesso (un metadato è un dato che, a sua volta, ne descrive un altro): allo scopo si utilizza il tag `<meta>`. Per esempio, quest'ultimo è utilizzato per definire la codifica dei caratteri in uso nel documento. Ciò che cambia rispetto alle precedenti versioni del linguaggio è la sintassi. Fino alla versione 4.01 si è usata la forma indicata nel Listato 2.4:

## **LISTATO 2.4** Elemento <meta> per la decodifica dei caratteri (HTML 4.01 e versioni precedenti)

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
```

Con HTML5 si può utilizzare una forma più breve (quella indicata nel Listato 2.4 è comunque riconosciuta):

## **LISTATO 2.5** Elemento <meta> per la decodifica dei caratteri (HTML5)

```
<meta charset="UTF-8">
```

Ancora una volta, otteniamo lo stesso risultato con una sintassi più agile e semplice da ricordare.

### Perché tanta enfasi sulla codifica dei caratteri?

Si immagini di aver riportato una citazione di Jim Morrison (il cantante del gruppo The Doors): "Il genere più importante di libertà è di essere ciò che si è davvero" e vedere scritto "Il genere più importante di libertà è di essere ciò che si è davvero". Non è solo la poesia della frase a venir meno! Sempre rifacendoci alla citazione di Morrison, se volessimo scrivere la frase con il ricorso ai caratteri speciali da adottare per ciascuna lettera accentata, otterremmo qualcosa del tipo: "Il genere più importante di libertà è di essere ciò che si è davvero". Otterremmo cioè un testo più lungo, la cui manutenzione risulta più onerosa. Pubblicare un testo avendo cura di selezionare la codifica appropriata consente di ridurre l'uso dei caratteri speciali. Il caso più frequente in cui capiterà di dover ricorrere a tali entità sarà dettato dalla necessità di trattare quei caratteri che sono anche parte della sintassi HTML5, come <, > ed &.

Un ultimo appunto in relazione all'elemento <meta> per la codifica dei caratteri. Poiché questa dichiarazione deve essere serializzata entro i primi 512 byte del file HTML, prendete la buona abitudine di inserire questo elemento subito dopo l'apertura dell'elemento di intestazione, possibilmente prima ancora di indicare il titolo del documento.

### Il mistero dei 512 byte!

In assenza di esplicite indicazioni relative alla codifica dei caratteri, il browser si sostituirà all'autore del documento nella scelta di una codifica, optando per quella predefinita. Questo accade se il browser, dopo aver letto i primi byte del documento, in genere appunto 512, che sono caricati in una memoria tampone (in gergo si dice che sono "bufferizzati"), non trova una dichiarazione pertinente. Se questa dichiarazione è posta oltre tale soglia, è probabile che sia necessario rileggere la pagina web o parte dei file a essa collegati per analizzarne nuovamente il contenuto alla luce della codifica "tardivamente" impostata. Peggio ancora, si è esposti a una vulnerabilità legata al tentativo di Internet Explorer di "indovinare" la codifica dei caratteri.

## Elemento <header>

Secondo la specifica, questo elemento contrassegna l'intestazione di una sezione del documento. Di norma contiene elementi di carattere introduttivo o di supporto alla navigazione. Poiché ogni sezione può essere dotata di una propria intestazione, nello stesso documento possono esistere più intestazioni: si pensi per esempio a un blog, che avrà un'intestazione generale in cui sono racchiusi il logo (se presente), il nome del blog, eventualmente un suo slogan, la cosiddetta tagline e una serie di collegamenti ipertestuali per la navigazione nel sito. È probabile che la home page contenga i primi paragrafi degli articoli più recenti. Ciascuno di essi avrà una propria intestazione, contenente il titolo dell'articolo (che potrebbe essere esso stesso un link alla versione integrale dell'articolo presentandosi quale supporto alla navigazione come suggerito dalla specifica), il nome dell'autore, la data e l'ora di pubblicazione.

#### **LISTATO 2.6** Esempio: elemento <header>

```
<header>
  <h1><a href="open-data-intro.html">Open Data libera le informazioni</a></h1>
  <p>Scritto da Gabriele Gigliotti</p>
  <p>Pubblicato il <time pubdate datetime="2010-10-22T15:30+01:00">22-10-2010</time>.</p>
</header>
```

L'intestazione racchiude: il titolo dell'articolo, con link alla versione integrale dello stesso, il nome dell'autore e la data di pubblicazione, su cui torneremo più avanti per introdurre il tag <time>.

## Elemento <nav>

Si è scritto che l'intestazione di una sezione può contenere elementi di supporto alla navigazione. Se presenti, questi elementi sono racchiusi in un tag <nav>. Non ha importanza che questi link puntino ad aree diverse della stessa pagina, pagine diverse dello stesso sito o esterne a esso. È invece essenziale che questi collegamenti rappresentino un ausilio alla navigazione. Non è immediato che un qualunque gruppo di link debba essere automaticamente demarcato in questo modo.

#### **LISTATO 2.7** Link di navigazione

```
<nav>
  <ul>
    <li><a href="altri_articoli.html">Archivio</a></li>
    <li><a href="ultimo_articolo.html">Ultimo Articolo</a></li>
    <li><a href="faq.html">Domande e Risposte (FAQ)</a></li>
  </ul>
</nav>
```

## Elemento <footer>

Così come per l'intestazione, anche per il piè di pagina possono esistere occorrenze multiple in una pagina. Attenzione, però: il piè di pagina potrebbe anche non essere tale! Il nome in effetti trae in inganno. Non esiste alcuna implicazione sulla disposizione di questa sezione in termini di layout. In altri termini, non è detto che il footer debba sempre trovare posto in fondo alla sezione in cui opera, anche se questo è quanto accade nella maggior parte dei casi.

### LISTATO 2.8 Un esempio di footer di un sito

```
<footer>
  <nav>
    <ul>
      <li><a href="/chisiamo.html">Chi siamo</a></li>
      <li><a href="/contatti.html">Come contattarci</a></li>
      <li><a href="/mappa-sito.html">Mappa del sito</a></li>
    </ul>
  </nav>
  <p>Copyright © 2010 Acme S.p.A.</p>
</footer>
```

## Elemento <section>

Il tag <section> demarca un insieme di contenuti tra loro logicamente correlati. La specifica fa riferimento in modo esplicito ad alcuni casi d'uso: il capitolo di un libro, le sezioni da cui è composta una tesi, le sezioni di una homepage (contatti, news ecc.).

Il significato di questo tag sarà più evidente quando si saranno passati in rassegna anche gli altri elementi di struttura. Esso, comunque, non è da intendersi come un contenitore generico privo di significato, utilizzato solo per applicare una serie di stili o come obiettivo di codice di scripting. In tal caso rimane sempre valido l'uso dell'elemento <div>. Inoltre, non si presta per un contenuto reso disponibile via RSS.

Ricapitolando, è bene non ricorrere a questo tag:

- per soli motivi di stile o di scripting da attribuire a una certa area del documento: in questi casi il tag <div> assolve meglio questo compito;
- quando l'area che si intende contrassegnare è messa a disposizione via RSS; in tal caso il tag <article>, introdotto anch'esso con questa versione del linguaggio, si presta meglio allo scopo.

## Elemento <article>

Si usa il tag <article> per marcare un contenuto che sia, almeno in linea di principio,

utilizzabile o distribuibile in modo indipendente dal resto del documento o anche solo della sezione in cui è racchiuso. Esempi tipici di questi contenuti possono essere l'articolo di un quotidiano online, la pubblicazione di un messaggio in un forum o presso un blog o, ancora, il commento pubblicato da un utente. I tag `<article>` possono essere annidati l'uno nell'altro: si pensi per esempio all'elemento più esterno come quello usato per contrassegnare l'articolo di un blog all'interno del quale sono inseriti i commenti degli utenti, ciascuno demarcato a sua volta da un tag `<article>`.

## ATTENZIONE

L'inserimento di più tag `<article>` l'uno dentro l'altro ha senso solo se esiste una qualche relazione tra di essi.

### LISTATO 2.9 Un articolo in HTML5

```
<article>
  <a href="open-data-intro.html">
    <header>
      <h1>Open Data libera le informazioni</h1>
      <p>Scritto da Gabriele Gigliotti</p>
      <p>Pubblicato il <time pubdate datetime="2010-10-22T15:30+01:00">22-10-2010</time>.</p>
    </header>
  </a>
  <p>Libero accesso ai dati (grezzi). In questo slogan si racchiude il concetto alla base dell'Open Data". Questa libertà presenta due declinazioni:</p>
  <ul>
    <li>rimozione dei vincoli posti da diritti d'autore e, in generale, da licenze che limitino l'accesso ai dati e al loro impiego;</li>
    <li>fruizione dei dati attraverso formati non proprietari che agevolino il riutilizzo degli stessi.</li>
  </ul>
  <aside>
    <p>"Quali sono i dati di cui si chiede a gran voce la circolazione senza restrizioni?"</p>
  </aside>
  <p>Nel 2009, sir Tim Berners-Lee ha tenuto una sessione al <a href="http://www.ted.com/"><abbr title="Technology Entertainment and Design">TED</abbr></a> (una serie di conferenze promosse, ogni anno, da una associazione non-profit), invocando "Dati grezzi subito!" (Raw Data Now!). Non è un caso che l'inventore del web si sia fatto portavoce di questa posizione. Grazie alla rete, quindi anche al web, che della rete è parte importante, sono stati abbattuti i costi di diffusione di quei dati che, per lungo tempo, si è chiesto di rendere pubblicamente disponibili.</p>
  <p>Ma quali sono i dati di cui si chiede a gran voce la circolazione senza restrizioni? <a href="open-data-intro.html" title="vai alla versione integrale dell'articolo">[vai all'articolo]</a> </p>
</article>
```

## Elemento <aside>

L'elemento `<aside>` marca un contenuto che è in relazione con l'elemento in cui `<aside>` è annidato, identificando tuttavia una relazione debole. Per esempio, se il tag viene usato nell'ambito di un elemento `<article>`, come nel Listato 2.9, si può dire che le informazioni contrassegnate con `<aside>`, se rimosse, non incidono negativamente sulla completezza dell'articolo, poiché il tag `<aside>` si limita ad arricchirlo con contenuti che sono solo "tangenzialmente" (avverbio usato nella specifica) correlati. `<aside>` può essere utilizzato, come nel codice sopra menzionato, per riportare il virgolettato di un testo, per aggiungere alcune note a margine o, infine, per introdurre uno spazio pubblicitario.

## L'articolo e i suoi fogli di stile

Volendo proporre una veste grafica che, seppure spartana, si presti a essere interpretata da browser vecchi e nuovi in modo ugualmente (s)gradevole, si proceda applicando un semplice foglio di stile:

### **LISTATO 2.10** Regole del foglio di stile

```
article, header, aside, footer { display: block; }
aside { float: right; margin: 5px; width: 30%; font-size: 1.3em; }
* { font-family: Verdana, sans-serif; }
h1 { font-size: 1.3em; }
a:link { color: #3a768a; }
a:visited { color: #67696b; }
a:hover { color: #3b86ca; }
```

La prima regola è necessaria per conferire ai nuovi elementi di struttura il comportamento di tag a livello blocco, ossia quei tag che creano una discontinuità nella pagina prima e dopo di essi, come `<p>`, `<h1>-<h6>`, `<div>` ecc.

### **NOTA**

Associare agli elementi di struttura a una visualizzazione a blocco permette di ottenere una gradevole rappresentazione anche nei browser che non supportano gli elementi, a eccezione delle versioni di Internet Explorer precedenti alla versione 9.

La seconda regola è volta solo a enfatizzare un virgolettato del testo come spesso accade nei quotidiani, dove alcune frasi sono riportate con un corpo più grande, di fianco all'articolo, proprio con l'obiettivo di attirare l'attenzione.

Le regole successive hanno il solo scopo di rendere più accettabile la visualizzazione del testo nel suo complesso.

Le versioni di Internet Explorer precedenti alla 9 non sono in grado di applicare uno stile a elementi sconosciuti. Ciò comporta la mancata applicazione delle prime due regole di stile e, nel complesso, un disastro nella visualizzazione del documento. Sembra che il solo modo per ovviare a questo sfacelo sia di creare in memoria questi elementi, via JavaScript, ricorrendo al metodo `createElement` dell'oggetto `document`.

**LISTATO 2.11** Creare gli elementi di struttura in memoria via JavaScript abilita le regole di stile in IE per le versioni meno recenti

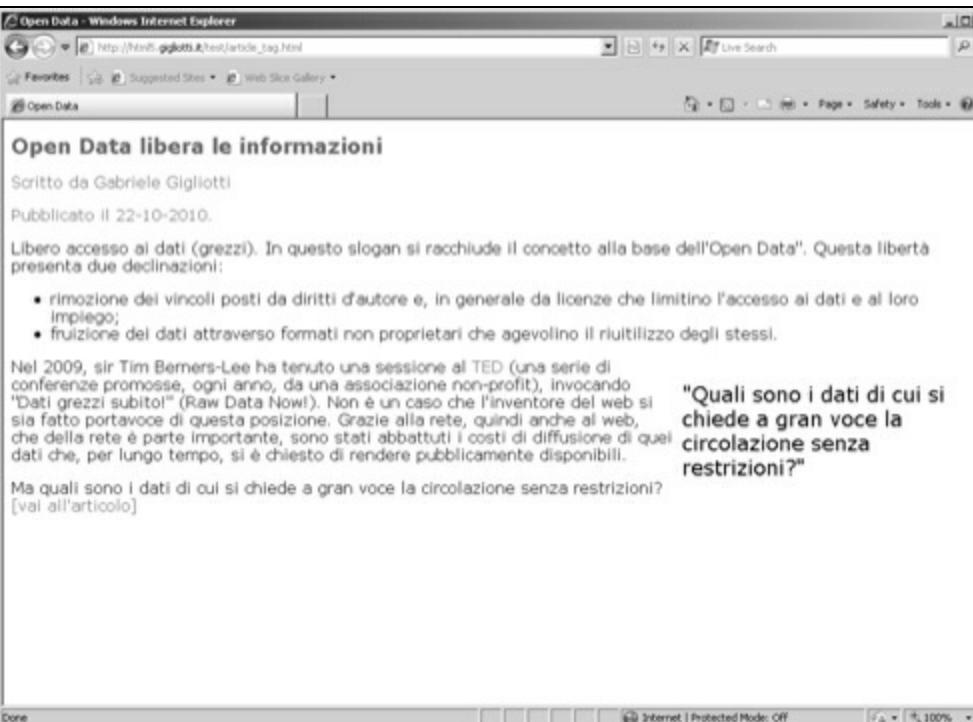
```
<script>  
document.createElement("article");  
document.createElement("header");  
document.createElement("aside");  
document.createElement("footer");  
</script>
```

È sufficiente creare un tipo di elemento una volta sola per documento, indipendentemente dal numero di volte in cui esso appare.

#### NOTA

È il caso di notare che questo approccio, benché risolva il problema della rappresentazione grafica, crea una dipendenza da JavaScript.

Si sono diffuse librerie JavaScript che fanno esclusivamente questo lavoro; una tra le più note è `html5shiv` (<http://html5shim.googlecode.com/svn/trunk/html5.js>). Essa può essere scaricata e inclusa nella propria pagina web avendo cura di usare commenti condizionali che consentano di accedere alla libreria solo alle versioni di Internet Explorer antecedenti alla 9.



**FIGURA 2.1** Un articolo scritto usando gli elementi di struttura di HTML5 così come viene visualizzato da IE8 dopo aver adottato specifici accorgimenti di visualizzazione.

### **LISTATO 2.12** Come includere html5shiv nelle proprie pagine web

```
<!--[if lt IE 9]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

## Nuovi elementi

Oltre ai tag che ridefiniscono, semplificandola, la struttura di un documento HTML, ne esistono di nuovi volti a potenziare la definizione dei contenuti. Di seguito si presenta un elenco non esaustivo di elementi che si possono già utilizzare.

### Elementi `<figure>` e `<figcaption>`

La specifica definisce i termini d'uso del tag `<figure>` impiegato per contrassegnare "illustrazioni, diagrammi, foto, listati di codice ecc. che sono citati nel testo principale ma che, senza alterarne il senso, possono essere rimossi dal contesto primario, per essere posti in un'altra area della stessa pagina, in pagine dedicate o in un'appendice". Pur se non espressamente indicati dalla specifica, anche file audio o video si prestano a essere marcati da questo elemento. L'insieme di informazioni racchiuse nel tag `<figure>` può essere arricchito da una didascalia. È qui che il tag `<figcaption>` entra in gioco. Ecco un esempio:

### **LISTATO 2.13** `<figure>` e `<figcaption>`: un esempio

```
<figure>
  
  <figcaption>Veduta esterna del Museo della Scienza di Valencia progettato dall'architetto Calatrava</figcaption>
</figure>
```

## Pillola di buon senso (parte I)

Potreste aver notato in questo caso una certa sovrapposizione tra l'attributo `alt`, che offre del testo alternativo nel caso in cui l'immagine non sia visualizzata (ma lo stesso testo appare come suggerimento anche in presenza dell'immagine) e l'elemento `<figcaption>` che fornisce una didascalia alla foto. Avete comprato questo manuale, sono in debito e, per dimostrarvi la mia gratitudine, non vi rovinerò la giornata discettando delle sottili differenze tra testo da fornire in alternativa alla foto e testo da presentare nella didascalia della foto (se voleste infliggervi da soli queste pene, potreste sempre trovare nella sezione 4.8.1.1.10 "A key part of the content" pane per i vostri denti). In casi del genere suggerisco di rifarsi al proprio buon senso rispondendo a una semplice domanda: un utente che ha scelto di disabilitare la visualizzazione delle immagini o un non vedente che usa uno screen reader ottiene una descrizione utile alla comprensione dell'immagine? Se la risposta è sì, avete assolto al compito di rendere il dato accessibile anche se dal punto di vista puramente semantico potreste aver scelto il posto meno ispirato in cui inserire il supporto testuale.

## Elemento `<hgroup>`

`<hgroup>` è il tag che raggruppa due o più intestazioni successive; ci si riferisce agli elementi di intestazione dal primo al sesto livello contrassegnati rispettivamente con gli elementi che vanno da `<h1>` a `<h6>`. Tutte le intestazioni inserite in una coppia `<hgroup> ... </hgroup>` sono trattate al livello del titolo più importante.

### **LISTATO 2.14** Raggruppare le intestazioni

```
<hgroup>
  <h1>Sideways</h1>
  <h2>In viaggio con Jack</h2>
</hgroup>
```

Le due intestazioni sono considerate come un tutt'uno e trattate come un'unica intestazione di primo livello: l'intestazione più importante tra le due presenti. Lo scopo di `<hgroup>`, nel caso specifico, è di nascondere l'esistenza di un titolo di secondo livello in modo che questo non interferisca con altri titoli `<h2>` che possono essere impiegati altrove, nello stesso documento, per definire un preciso schema logico. Nel Listato 2.14 infatti accade che il testo `In viaggio con Jack` possa considerarsi piuttosto un sottotitolo, ossia qualcosa di diverso da un titolo di secondo livello a tutti gli effetti.

## Elemento `<time>`

L'elemento demarca data e ora codificandole in modo leggibile da una macchina e proponendo, al contempo, una visualizzazione leggibile dall'utente.

## **LISTATO 2.15** Come contrassegnare l'informazione data/ora

```
<time datetime="2010-10-30T11:20:00Z" pubdate>Sabato 30 Ottobre 2010</time>
```

L'attributo `datetime` riporta la data e l'ora in formato ISO, ossia secondo il formato descritto in Figura 2.2.

L'indicatore del fuso orario posto al termine della stringa rappresentante data/ora è opzionale; se non viene specificato, l'ora sarà determinata dal software impiegato per visualizzare il documento HTML.

Anche l'attributo `datetime` è opzionale; se è assente, la data è determinata in base alla stringa posta all'interno del tag, in tal caso questa stringa deve essere una rappresentazione di data/ora conforme al formato ISO.

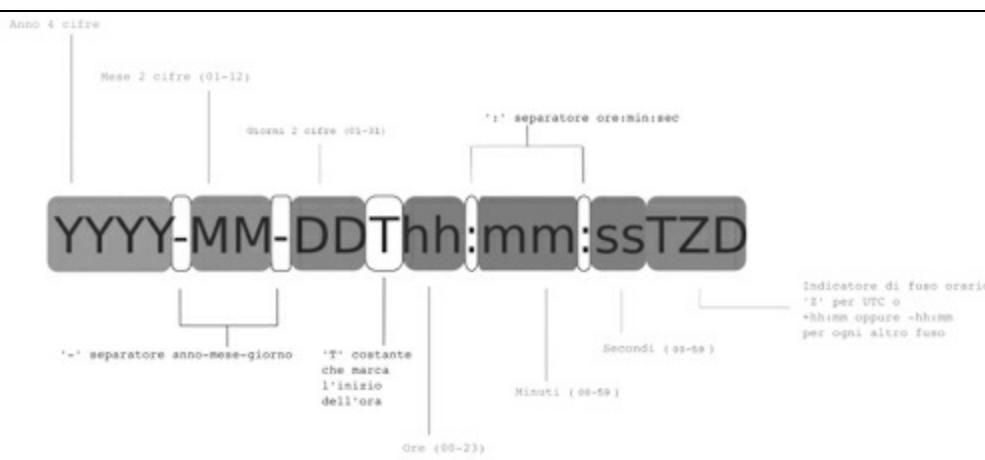
Ecco alcuni esempi d'uso del tag `time` per definire la data del 21/10/2010.

## **LISTATO 2.16** Marcatore `<time>`: alcuni esempi di rappresentazione della data 21/10/2010

```
<p>Era un <time datetime="2010-10-21">giovedì</time> di tardo ottobre quando ...</p>
```

```
<p>Oggi <time datetime="2010-10-21">21 ottobre 2010</time> ...</p>
```

```
<p>Erano solo le <time datetime="2010-10-21T05:00+01:00">5 di mattina</time> quando ho raggiunto Milano.</p>
```



**FIGURA 2.2** Elementi del formato ISO data/ora.

L'attributo booleano `pubdate` stabilisce che la data impostata è quella di pubblicazione dell'articolo in cui l'elemento `<time>` è innestato. Se non esiste alcun tag `<article>` all'interno del quale si trovi il tag `<time>`, vuol dire che la data di pubblicazione si riferisce all'intero documento.

## Elementi `<ruby>`, `<rt>`, `<rp>`

Tre nuovi elementi sono stati concepiti esclusivamente per il supporto alle lingue

ideografiche orientali. Ruby è il termine generico usato per la rappresentazione fonetica di tali lingue, o almeno del cinese e del giapponese: in quest'ultima, nello specifico, viene chiamato di solito furigana. I furigana o ruby svolgono una funzione di ausilio alla pronuncia dei caratteri ideografici cui si riferiscono. Tali caratteri sono chiamati, in giapponese, Kanji. Sono, dunque, impiegati nel caso di termini poco noti o che richiedano una pronuncia diversa da quella abituale o, come nel caso che prendiamo a esempio per questo paragrafo, quando il testo è rivolto a chi sta apprendendo la lingua e non è detto ne conosca la pronuncia.

A puro titolo di esempio prendiamo in esame la seguente frase:

**LISTATO 2.17** Lo sto scrivendo in giapponese, scritto in... giapponese

私は日本語で書いています。

Se volessimo applicare a questi ideogrammi i relativi caratteri furigana otterremmo una rappresentazione di questo tipo:

わたし にほんご か  
私は日本語で書いています。

**FIGURA 2.3** Kanji e furigana.



**FIGURA 2.4** I caratteri ruby sono posti al di sopra o immediatamente a destra del carattere di cui illustrano la pronuncia.

Per rendere più evidente le sue componenti possiamo rifarcirsi alla Figura 2.4, in cui sono chiaramente distinte le due componenti.

I furigana sono posti al di sopra dei rispettivi ideogrammi. In giapponese non tutti i caratteri richiedono la pronuncia in furigana. Inoltre, si mettono su gruppi di 1, 2 o 3 ideogrammi. Se decidiamo di annotare in modo opportuno questa frase in modo che i simboli di rappresentazione fonetica appaiano in corrispondenza degli ideogrammi di cui

sono chiamati a facilitare la pronuncia, mettiamo per prima cosa in sequenza i Kanji con i relativi caratteri furigana ottenendo quanto illustrato nel Listato 2.18:

#### **LISTATO 2.18** Kanji e furigana

私わたしは日本語にほんごで書かいています。

Su sfondo nero sono rappresentati gli ideogrammi, mentre i furigana a essi relativi sono stati evidenziati in grigio. Per applicare gli elementi HTML si procede come segue:

#### **LISTATO 2.19** <ruby> marca ideogrammi e annotazioni ruby

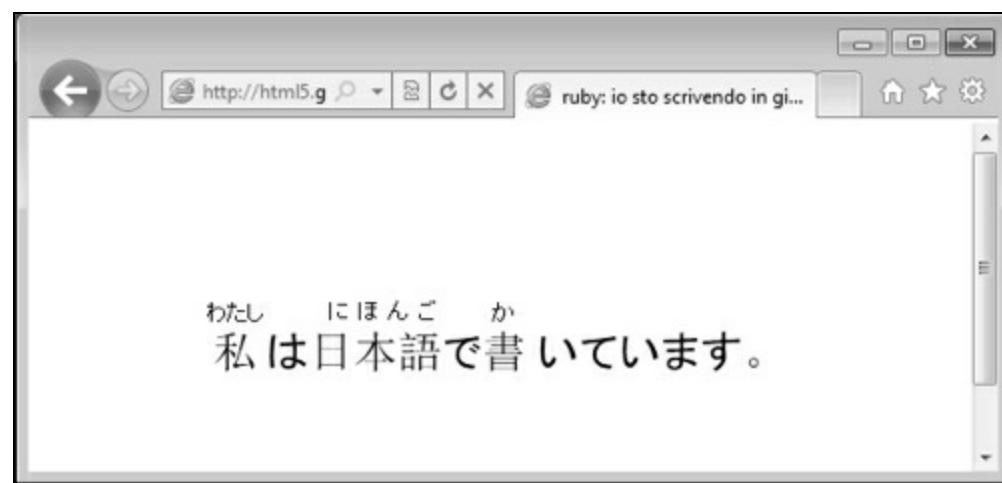
```
<ruby>uno o più ideogrammi oggetto di annotazione<rt>ruby (o furigana) relativi all'ideogramma o agli ideogrammi  
di cui illustrano la pronuncia</ruby>
```

Applicando questa regola alla nostra frase avremo:

#### **LISTATO 2.20** Ideogrammi e relativi furigana contrassegnati con elementi HTML

```
<ruby>私<rt>わたし</rt></ruby>は<ruby>日本語<rt>にほんご</rt></ruby>で<ruby>書<rt>か</rt></ruby>いています。
```

All'interno dell'elemento `<ruby>` si innesta il tag `<rt>`, acronimo di Ruby Text, usato per contrassegnare gli ideogrammi con annotazioni ruby. Ed ecco, in Figura 2.5, il risultato prodotto da IE9.



**FIGURA 2.5** IE9 beta supporta gli elementi `<ruby>` e `<rt>`.

Sfortunatamente, se il browser non supporta questi elementi le annotazioni ruby non sono mostrate come in Figura 2.5 ma appaiono di fianco agli ideogrammi, confondendosi con essi e compromettendo la leggibilità della frase. Ecco che cosa accade in Firefox 4, anch'esso in versione beta (Figura 2.6).

## NOTA

Firefox interpreta correttamente le annotazioni ruby se si ha cura di installare un apposito componente disponibile al seguente indirizzo: <https://addons.mozilla.org/en-US/firefox/addon/1935/>.

Per ovviare a questo inconveniente, riscriviamo la frase mettendo i caratteri ruby tra parentesi tonda in modo da avere:

### LISTATO 2.21 Annotazione ruby tra parentesi

私(わたし)は日本語(にほんご)で書(か)いています。



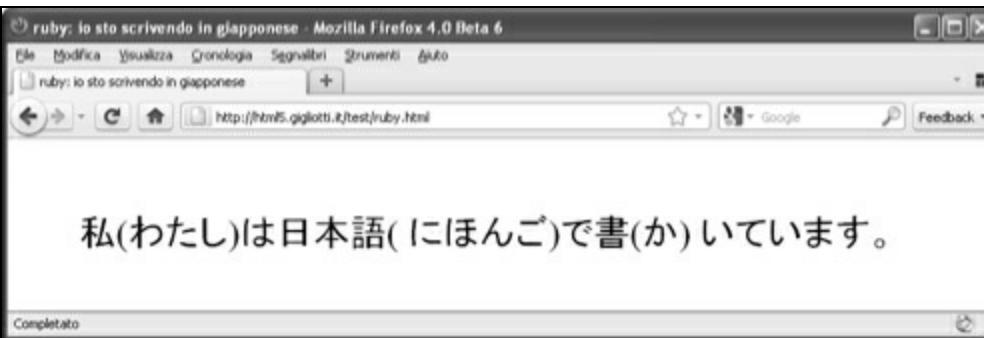
**FIGURA 2.6** Il mancato riconoscimento degli elementi porta a un appiattimento dei caratteri, la rappresentazione ne risulta compromessa.

Quindi usiamo il tag `<rp> ... </rp>` al solo scopo di contrassegnare le parentesi. `rp` è infatti l'acronimo di Ruby Parenthesis, ossia coppia di parentesi tonde usate per circoscrivere le annotazioni ruby. Applicando tutti e tre questi elementi per contrassegnare la frase otteniamo il codice definitivo:

### LISTATO 2.22 Annotazione con supporto per browser che non riconoscono gli elementi

```
<ruby>私<rp>(</rp><rt>わたし</rt><rp>)</rp></ruby>は<ruby>日本語<rp>(</rp><rt>にほんご</rt><rp>)</rp></ruby>で<ruby>書<rp>(</rp><rt>か</rt><rp>)</rp></ruby>いています。
```

Per i browser che, come IE9, riconoscono e interpretano correttamente il codice, le parentesi sono trattate come elemento superfluo ai fini della rappresentazione e dunque sono ignorate. Ciò significa che IE9 mostrerà all'utente le stesse informazioni visualizzate nella Figura 2.5 anche se il codice di marcatura è diverso. Per contro, Firefox 4, ancora indietro su questo punto, mostrerà anche le parentesi, permettendo così di distinguere in modo chiaro gli ideogrammi dalle annotazioni ruby (Figura 2.7).



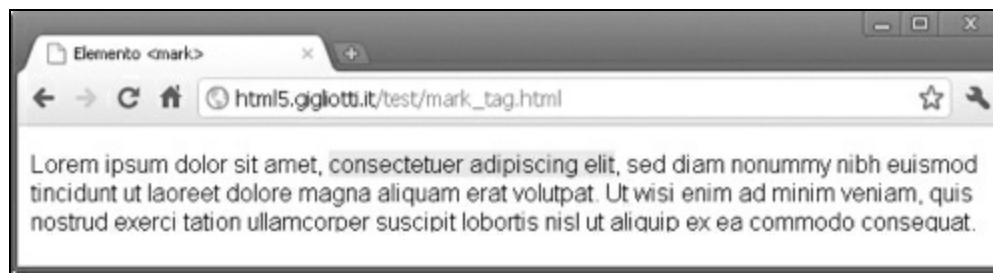
**FIGURA 2.7** Le parentesi aiutano a distinguere gli ideogrammi dalle annotazioni ruby.

## Elementi `<progress>` e `<meter>`

Ecco un caso in cui la definizione del significato degli elementi può portare allo stallo. Sia `<progress>` sia `<meter>` dovrebbero essere resi come barre di avanzamento: come comprendere qual è il tag corretto da usare? L'elemento `<meter>` rappresenta una grandezza scalare all'interno di un intervallo noto: per esempio lo spazio su disco, un indice di popolarità come il Google rank, un indicatore di rilevanza. L'elemento `<progress>` indica il grado di completamento di un'attività: per esempio il caricamento di un file. La percentuale di completamento potrebbe non essere nota, in tal caso il browser dovrebbe mostrare un indicatore diverso dalla barra di completamento per segnalare che l'attività è in corso di esecuzione ma il tempo necessario per giungere al termine della stessa non è noto.

## Elemento `<mark>`

`<mark>` annota una parte di testo per enfatizzarne il significato. I browser che riconoscono e interpretano correttamente questo elemento ne propongono il testo come se fosse evidenziato (Figura 2.8).



**FIGURA 2.8** Chrome visualizza su sfondo giallo il testo contrassegnato con `<mark>`.

La specifica suggerisce (una tra le tante possibili applicazioni) che l'elemento possa essere impiegato nella pagina dei risultati, ottenuta a fronte di una ricerca, per demarcare la o le parole chiave inserite dall'utente.

<p>Lorem ipsum dolor sit amet, <mark>consectetuer adipiscing elit</mark>, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.</p>

## Vecchi non più vecchi...

Alcuni elementi continuano a fare parte della specifica, seppure con un significato diverso rispetto a quello originario.

### Elemento <a>

Alcune novità riguardano anche un elemento così importante come quello alla base dei collegamenti ipertestuali. La specifica HTML5 introduce due nuovi attributi.

- **ping**: permette di impostare uno o più URL che verranno contattati se l'utente deciderà di seguire il collegamento ipertestuale. L'impiego più immediato è quello del tracciamento degli utenti.
- **media**: accetta una stringa che identifica il tipo di supporto per il quale sia stato concepito il documento cui punta il collegamento. I possibili valori sono nove: 'all', 'aural', 'braille', 'handheld', 'print', 'projection', 'screen', 'tty', 'tv', 'all' è il valore predefinito se non ne viene espressamente indicato uno diverso. A oggi questo attributo ha solo funzione informativa.

Inoltre si stabilisce che sia possibile usare un elemento <a> anche per "un intero paragrafo, liste, tabelle persino intere sezioni di documento, fintantoché non contengono contenuto interattivo (come, per esempio, pulsanti o altri link)". Questo non era possibile nelle precedenti versioni del linguaggio.

Se avete osservato con cura il listato proposto per illustrare l'utilizzo del tag <article> avrete forse scoperto che tutto il tag <aside>, e dunque anche tutto il suo contenuto (un titolo di primo livello e due paragrafi), sono racchiusi in un tag <a>. Un caso d'utilizzo è quello del quotidiano che riporta in prima pagina titolo e una foto, laddove entrambi gli elementi sono contrassegnati con il medesimo link di richiamo all'articolo. Ecco un esempio preso dal sito del Corriere della Sera (<http://www.corriere.it>) sull'articolo di apertura di sabato 11 settembre 2010.

#### **LISTATO 2.24** Impaginazione di articolo in home page (XHTML 1.0 Transitional)

```
<h1>
<a href="/cronache/10_settembre_11/capua-operai-morti_d9742098-bd87-11df-bf84-00144f02aabe.shtml">Incidenti sul
lavoro, 4 morti<br/>Napolitano: sono indignato</a>
</h1>
```

```
<a href="/cronache/10_settembre_11/capua-operai-morti_d9742098-bd87-11df-bf84-00144f02aabe.shtml">
</a>
<p><!-- segue testo di richiamo all'articolo vero e proprio --></p>
```

Questo listato produce il risultato apprezzabile in Figura 2.9. Sfruttando la possibilità offerta da HTML5 di utilizzare un collegamento ipertestuale che possa contrassegnare un'intera sezione (marcata da `<header>...</header>`) segue che questa titolazione potrebbe essere riscritta come proposto nel Listato 2.25.

## **LISTATO 2.25** Impaginazione di articolo in home page (HTML5)

```
<article>
  <a href="/cronache/10_settembre_11/capua-operai-morti_d9742098-bd87-11df-bf84-00144f02aabe.shtml">
    <header>
      <h1>Incidenti sul lavoro, 4 morti<br/>Napolitano: sono indignato</h1>
      
    </header>
  </a>
  <p> <!-- segue sommario dell'articolo vero e proprio -->
</p>
</article>
```

## Elemento `<address>`

Nella nuova versione del linguaggio lo scopo dell'elemento `<address>` è di proporre informazioni di contatto relative all'articolo, se immediatamente contenuto in un tag `<article>`, o all'intera pagina, se il suo contenitore più diretto è un tag `<body>`.

# Incidenti sul lavoro, 4 morti Napolitano: sono indignato



## FLASHI

22:50 | SI

Europei, It

22:43 | SI

Cagliari-R

22:40 | SI

fermato da

22:35 | CI

Mutilazioni

## PIU' lett

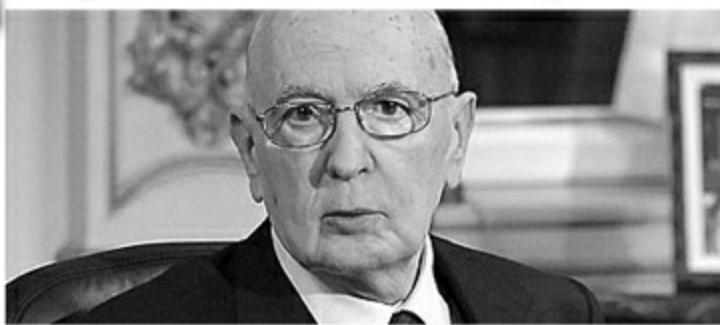
Tragedia sul lavoro,

Foto|Video

22:33 | CRONACHE | Il Capo dello Stato: si ripetono incidenti mortali

www.corriere.it/cronache/10\_settembre\_11/capua-operai-morti\_d9742098-bd87-11df-bf84-00144f02aabe.shtml

# Incidenti sul lavoro, 4 morti Napolitano: sono indignato



## FLASHI

22:27 | E

lavoro: Epi

22:23 | CI

Prestigiac  
piano suo

22:00 | SI

Madrid-Os

## PIU' lett

## ZAPPIN

22:33 | CRONACHE | Il Capo dello Stato: si ripetono incidenti mortali

www.corriere.it/cronache/10\_settembre\_11/capua-operai-morti\_d9742098-bd87-11df-bf84-00144f02aabe.shtml

**FIGURA 2.9** Sia il titolo sia l'immagine sono contrassegnati da due elementi `<a>`: entrambi puntano allo stesso documento.

Ed ecco ora la parte che genera confusione: non è corretto marcare acriticamente un qualunque indirizzo che appaia nel documento con `<address>`. Ciò deve avvenire solo quando questi dettagli sono utili a contattare l'autore dell'articolo o più in generale chi gestisce i contenuti di documento. In questo senso si parla di informazioni di contatto in senso lato. Non è necessario che esse rappresentino sempre un indirizzo postale, è sufficiente si tratti di un dettaglio utile a stabilire un contatto (e-mail, telefono ecc.).

È invece l'elemento di paragrafo `<p>` che deve essere usato in tutti quei casi in cui questa relazione non esiste perché si sta marcando un indirizzo generico.

Un esempio aiuterà a comprendere meglio quando utilizzare questo elemento.

**LISTATO 2.26** Quando usare l'elemento `<address>`: un caso concreto

```

<article>
  <header>
    <h1><a href="open-data-intro.html" title="vai alla versione integrale dell'articolo">Senso dell'orientamento: questo sconosciuto</a></h1>
    <p>Scritto da Gabriele Gigliotti</p>
    <p>Pubblicato il <time pubdate datetime="2010-09-12T15:30+01:00">12-09-2010</time>.</p>
  </header>
  <p>&quot;Dove diavolo si trova Via Solferino, 28!&quot;. Ancora una volta mi trovavo a 15 minuti da un importante colloquio di lavoro senza avere la benché minima idea di dove fossi rispetto alla mia meta! Il senso dell'orientamento mi ha sempre fatto difetto, per fortuna questa volta avevo con me il mio smartphone e le mappe di Google mi hanno aiutato a raggiungere l'indirizzo giusto in tempo.</p>
  <p>Mi presento dunque come al mio solito: trafilato e sorridente come Gebrselassie al termine della maratona di Berlino ...<a href="short_story_01.html" title="vai alla versione integrale dell'articolo">[vai all'articolo]</a></p>
  <footer>
    <p>Autore informatico, appassionato di web e open data. </p>
    <address>
      Email: <a href="mailto:gabriele.gigliotti@gmail.com">gabriele.gigliotti@gmail.com</a><br>
      Twitter: <a href="http://twitter.com/ridillo">@ridillo</a>
    </address>
  </footer>
</article>

```

Nel testo dell'esempio appare "Via Solferino, 28". Pur essendo un indirizzo a tutti gli effetti, si tratta di parte del racconto che non fornisce alcun dettaglio utile a contattare l'autore dell'articolo o chi gestisce il documento web in cui l'articolo appare. Al contrario, l'indirizzo e-mail e l'account di Twitter sono riferimenti utili a prendere contatto con l'autore; per questo motivo è corretto che siano inclusi in un elemento `<address>`.

Dovrebbe, invece risultare palese, che sia scorretto utilizzare `<address>` per identificare qualunque altra informazione che non sia un'informazione di contatto:

#### **LISTATO 2.27** Uso improprio del tag `<address>`

```
<address>Last Modified: 1999/12/24 23:37:50</address>
```

## Pillola di buon senso (parte II)

Il solo vero validatore semantico esistente oggi è una persona. Non esiste alcun software in grado di stabilire se l'elemento `<address>` sia stato utilizzato in modo più o meno appropriato. Detto questo, dopo aver condotto una rapida analisi sulla necessità di usare o meno questo elemento, non ragioniam di lor, ma guarda e passa, come disse Dante nella sua Divina Commedia, cioè fa' la tua scelta e passa oltre senza ulteriori patemi. Ci sono altri aspetti certamente più critici su cui vorrete concentrare le vostre energie.

## Elemento `<b>`

“b” sta per bold, ossia grassetto. In effetti l’uso che si è fatto in passato di questo elemento era proprio relativo alla formattazione del testo. Con HTML5 buona parte degli elementi utilizzati con il solo obiettivo di trasmettere un’informazione di stile sono stati soppressi: il tag `<b>` è uno dei pochi di questa famiglia a sopravvivere. Il motivo per cui è uscito indenne da queste sforbicate è il suo diffuso impiego in miliardi di pagine web (lo stesso discorso vale per il tag `<i>`).

Secondo la specifica, l’elemento è utile a contrassegnare un vocabolo o una frase che devono essere evidenziati rispetto al testo ordinario; alcuni esempi possono essere l’incipit di un articolo o le parole chiave in un testo. È legittimo attendersi che la decisione di mantenere due elementi come `<b>` e `<strong>`, che presentano aree di sovrapposizione, possa confondere. Ciò perdura nonostante gli sforzi fatti per stabilire quando usare l’uno e quando l’altro. Personalmente sono solito definire l’elemento `<b>` come tag del “disperato”. Esso non è da utilizzare come titolo poiché per questo si usano i tag di titolo da `<h1>` a `<h6>`; se ne scoraggia l’uso per enfatizzare il testo perché qui si usa il tag `<em>`; è da non utilizzare per dare importanza al testo, dato che questo è compito del tag `<strong>`; di certo non può essere impiegato per evidenziare singole parole o frasi perché il tag `<mark>` gioca qui il suo ruolo. Al di fuori dei pochi esempi elencati sopra, tutti espressamente previsti dalla specifica, non riesco a pensare ad altri motivi che ne giustifichino l’impiego se non la retrocompatibilità con le vecchie pagine web.

## Elemento `<cite>`

Un altro caso di rielaborazione del significato originario ci è offerto dall’elemento `<cite>`, che ora ha un’interpretazione più restrittiva. Infatti, sebbene possa continuare come in passato a rappresentare un riferimento ad altre fonti, per esempio un libro o una canzone, non può più essere usato per contrassegnare il nome di una persona. Su questo punto sono nati dei contrasti tra chi è a favore di questa ridefinizione e chi, in omaggio al principio secondo cui alla prassi diffusa presso gli autori deve essere riconosciuta precedenza rispetto alla specifica, pretende che questo elemento possa essere utilizzato come già oggi accade per citare anche i nomi di persona. Fiumi d’inchostro, o sarebbe meglio dire di bit in questo caso, sono stati versati per stabilire quale sia il testo più appropriato da marcare con `<cite>`: la discussione è ancora viva e forse condurrà in futuro a un’ulteriore rivisitazione del significato di questo tag.

## Elemento `<hr>`

Le iniziali da cui è composto l’elemento (“hr” sta per Horizontal Rule, ossia riga orizzontale) ne tradiscono quel significato visuale che ora è venuto meno. In HTML5 `<hr>` è definito come tag da impiegare per impostare un’interruzione tematica. In altri termini, l’elemento segna il punto di passaggio tra un argomento che si conclude e un altro che inizia: per esempio, un cambio di scena in un copione.

## Elemento <i>

Per l'elemento <i> valgono le stesse considerazioni fatte per l'elemento <b>: la sua ubiquità lo ha salvato dall'estinzione! Anche qui vengono meno le implicazioni stilistiche che vogliono sempre in corsivo il testo contrassegnato con questo elemento. Secondo quanto espressamente previsto dalla specifica, scenari d'uso sono, tra gli altri, i termini tecnici e le frasi idiomatiche in lingua diversa dalla propria. In ogni caso si incoraggiano gli autori a utilizzare il tag <em> per contrassegnare frasi o singole parole che si desidera enfatizzare. Laddove ciò non sia possibile, si suggerisce di accompagnare il tag con l'attributo `class` in modo da fornire informazioni riguardo all'impiego dell'elemento (oltre che alla sua rappresentazione visuale). Questo consentirà in futuro di rivederne l'uso (magari sostituendolo con un elemento più appropriato) senza per questo dover passare di nuovo in rassegna tutti i diversi casi in cui esso è impiegato.

## Elemento <label>

Poco cambia per le etichette assegnate ai controlli da cui è composto un form. La specifica si limita a precisare che il comportamento dell'elemento <label> che consiste nel trasferire il focus dall'etichetta al controllo corrispondente, o addirittura interagire con esso (per esempio un clic sull'etichetta di una casella di controllo fa apparire automaticamente un segno di spunta all'interno della casella) deve venire meno se questo non è il comportamento predefinito della piattaforma su cui opera l'utente.

## Elemento <menu>

Ancora in HTML 4.01 questo elemento era stato contrassegnato come deprecato ma è stato ripristinato nella nuova versione del linguaggio. L'elemento è pensato per racchiudere una lista di comandi. Sono i valori dell'attributo `type` che aiutano a stabilire che tipo di menu sia stato dichiarato. I tre possibili valori per questo attributo sono:

- `context`: viene dichiarato un menu di tipo contestuale (un esempio di menu contestuale è quello che in Windows si ottiene facendo clic con il tasto destro del mouse);
- `toolbar`: identifica un menu di tipo "barra di strumenti";
- `list`: definisce una lista di comandi. Questo è il valore predefinito dell'attributo `type`. Ciò significa che se l'attributo non è esplicitamente impostato a un valore diverso, si assume appartenga a questo tipo.

L'attributo `label` è invece utilizzato per conferire un nome al menu.

## Elemento <small>

Finora utilizzato per riprodurre il testo in un corpo più piccolo di quello abituale, l'elemento <small> diventa ora il tag con cui si contrassegnano testi come le note legali, diritti d'autore e piccole note a margine. In ogni caso deve trattarsi di un testo breve. Se fosse troppo lungo, sarebbe necessario optare per un elemento diverso perché potrebbe trattarsi del contenuto principale.

## Elemento <strong>

Da elemento usato per contrassegnare del testo per attribuirvi una forte enfasi, diventa ora un elemento usato per identificare l'importanza di una parola o di una frase. Temo che in questo caso non possa che trattarsi di differenze così esiziali che per lo più saranno ignorate dagli autori.

## Elementi e attributi obsoleti

HTML5 nasce in stretta continuità con le versioni precedenti del linguaggio. Uno degli obiettivi di questa nuova versione è proprio di garantire un'adeguata visualizzazione delle centinaia di milioni di pagine web già pubblicate, pur offrendo un nuovo e più potente strumento. Detto ciò, com'era lecito aspettarsi, la specifica scoraggia l'impiego di una serie di elementi e attributi. Quando ciò accade è dovuto a uno dei motivi spiegati di seguito.

- Si vuole disincentivare l'uso di elementi e/o attributi il cui fine è esclusivamente stilistico (per esempio <big>, <blink>). Questo aspetto è demandato ai fogli di stile, il linguaggio di marcatura deve essere sgravato da questo fardello.
- L'elemento/attributo è dichiarato obsoleto perché la sua funzione è svolta da un altro elemento/attributo o non è più necessaria (per esempio <bgsound> è sostituito da <audio>, l'attributo `target` sull'elemento `link` non è più necessario e può essere omesso).
- L'elemento/attributo ha dimostrato di causare problemi di accessibilità o usabilità, o entrambi (per esempio <frame>, <frameset>, <noframes>: quale altra fine potevamo prospettare per il tag più odiato della storia di HTML? Esso è stato l'unico ad avere portato gestori di siti web in tutto il mondo ad aggiungere alle proprie pagine il logo "io odio i frame").

Segue la tabella di elementi dichiarati obsoleti:

**TABELLA 2.2** Elementi obsoleti non conformi

acronym	font	multicol	spacer
basefont	frame	nobr	strike
big	frameset	noembed	tt
bgsound	noframes	nextid	u
blink	isindex	plaintext	xmp
center	listing	rb	

Gli attributi dichiarati obsoleti sono all'incirca un centinaio (si veda la sezione 11.2 "Non-conforming features" della specifica).

## Deprecato vs obsoleto

Dismettere un elemento o un attributo, fino alla versione 4.01 del linguaggio, equivaleva a seguire questo schema:

- l'elemento/attributo viene dichiarato deprecato nella specifica;
- chiunque sviluppi pagine web (almeno i più avveduti tra questi) cessa di usarlo perché il supporto potrebbe venire meno in futuro;
- l'elemento/attributo deprecato è dichiarato obsoleto in una versione successiva della specifica;
- i browser, e più in generale tutti i programmi client che si collegano al Web, pongono fine al supporto.

Con HTML5 questo schema è stato messo in discussione e sostituito con un altro che identifica direttamente l'elemento/attributo come obsoleto (con una distinzione apparentemente controllata tra funzionalità obsolete non conformi e obsolete ma conformi). A questo proposito ho trovato divertente il modo spicchio ma ironico con cui Ian Hickson, editore della specifica HTML5, risponde sul canale irc `#whatwg` (estratto di una chat del 3 agosto 2009):

```
# [00:15] <annevk2> Hixie, perché mai non usiamo il concetto di "deprecato" ora che abbiamo i messaggi di avviso?
# [00:16] <Hixie> annevk2: "Deprecato" non è nel mio dizionario.
# [00:16] <Hixie> Voglio dire, letteralmente. Ho cercato "Deprecato" nel dizionario sul Mac OS X e non l'ho trovato.
```

## Riferimenti alle risorse citate e altri link utili

- L'articolo in cui è citata la vulnerabilità cui risulta esposto Internet Explorer in relazione alla codifica adottata nelle pagine web:

<http://code.google.com/p/doctype/wiki/ArticleUtf7>

- Ulteriori approfondimenti sul tema della codifica dei caratteri sono disponibili attraverso una serie di articoli di carattere divulgativo presso il sito del W3C al seguente indirizzo: <http://www.w3.org/International/articlelist#characters>
- L'articolo che illustra come servire gli elementi di struttura in modo che versioni di Internet Explorer precedenti alla versione 9 siano in grado di applicarvi le regole di stile definite: <http://blog.whatwg.org/supporting-new-elements-in-ie>
- Su <http://html5shim.googlecode.com/svn/trunk/html5.js> è disponibile la libreria JavaScript html5shiv: abilita vecchie versioni di Internet Explorer alla stilizzazione degli elementi. Attenzione alle potenziali cause di conflitto con la libreria Modernizr che utilizzeremo nel corso di tutto il manuale.
- La sezione della specifica numero 4.8.1.1.10 intitolata "A key part of the content" che discetta lungamente su come sia più semanticamente corretto annotare un'immagine con testo descrittivo è consultabile a questo indirizzo: <http://www.w3.org/TR/html5/embedded-content-1.html#a-key-part-of-the-content>
- Se il paragrafo su `<ruby>`, `<rt>`, `<rp>` ha sollecitato la vostra curiosità sul Giappone, <http://www.strayinjapan.it/> è una risorsa in lingua italiana che spazia sui temi legati alla cultura giapponese.
- Per un elenco completo degli attributi dichiarati obsoleti si consiglia di prendere visione della sezione 11.2 intitolata: "Non-conforming features": <http://www.w3.org/TR/html5/obsolete.html#non-conforming-features>
- La versione integrale della sessione via chat tenutasi sul canale irc #whatwg il 3 agosto 2009 dove l'editore della specifica HTML5, Ian Hickson, esprime il suo punto di vista sul tema "deprecato vs obsoleto": <http://krijnhoetmer.nl/irc-logs/whatwg/20090803>

## Conclusioni

In questo capitolo abbiamo visto come reimpostare la struttura di una pagina web migliorandone, al contempo, la semantica. Non si tratta di un puro esercizio di stile, dato che a trarne giovamento sono tempi e costi di manutenzione legati a un codice più comprensibile e meno verboso. Permane la difficoltà di capire in quali circostanze applicare un dato elemento in base al significato che gli è stato attribuito dalla specifica. Mettete pure in conto un periodo di rodaggio per prendere confidenza con i nuovi elementi e con quelli che hanno mutato di significato. Più di ogni altra cosa, non state troppo scrupolosi nella scelta dei tag. Esistono aree di sovrapposizione tra elementi diversi e anche nel caso ideale di regole più stringenti di quelle proposte dalla specifica, persone diverse sarebbero comunque arrivate a contrassegnare uno stesso documento in modo diverso. La sola cosa che conta è essere coerenti: scegliete un approccio e mantenetelo per lo sviluppo del vostro progetto. Questo è il solo modo che vi consente poi di ritornare sui vostri passi nel caso scopriate di aver commesso un errore o semplicemente vogliate aggiornare i vostri documenti.

I prossimi due capitoli sono dedicati ai form, un'area in cui HTML5 ha introdotto controlli così evoluti da fare impallidire quelli utilizzati sinora. Dunque, buona lettura con i form 2.0.

# Web Forms 2.0 (prima parte)

Molto prima che il Web 2.0 diventasse realtà, i form hanno garantito agli utenti una efficace, seppur rudimentale, modalità di interazione. Forse troppo rudimentale! HTML5 aggiunge agli attuali controlli 13 campi input, 1 elemento e 10 nuovi attributi. Temi dominanti di queste innovazioni sono: la semantica; la maggiore semplicità d'uso; l'accessibilità; la traduzione in linguaggio dichiarativo di quelle che, finora, sono state soluzioni di scripting (proposte attraverso librerie come jQuery, Dojo, Prototype) alle ben note limitazioni dei form.

**TABELLA 3.1** Supporto dei Web Forms 2.0

ELEMENTO SUPPORTATO	BROWSER
Web Forms 2.0	Chrome 5+*, Firefox 4.0+*, Opera 10.1+, Safari 4*

\* In questi browser l'implementazione è solo parziale.

## Campi input

La Tabella 3.1 evidenzia un supporto tutt'altro che maturo di questa parte della specifica. Tuttavia, un vantaggio comune dei nuovi campi input risiede proprio nella migliore definizione semantica del contenuto. Fino a oggi, che si chiedesse di inserire un numero di telefono, un indirizzo email o un URL si sarebbe utilizzata la stessa istruzione.

**LISTATO 3.1** Un campo di testo standard

```
<input type="text">
```

Il generico campo di testo usato per i fini più disparati può ora essere sostituito con controlli che offrono una migliore definizione dei dati, perché più specifici: il tipo `search` identifica il campo in cui immettere le chiavi di ricerca, il tipo `email` è usato per invitare l'utente a inserire uno o più indirizzi di posta elettronica, e così via. Ogni nuovo controllo contrassegna il tipo di dato in modo più preciso di quanto non fosse possibile ottenere in passato. Fino alla versione 4.01 del linguaggio esistevano solo 10 valori possibili per

l'attributo `input (button, checkbox, file, hidden, image, password, radio, reset, submit, text);` con le aggiunte della nuova specifica se ne contano 23. Se l'attributo `type` è omesso o se viene proposto un valore diverso da quelli riconosciuti dal browser, tutto ciò che si ottiene è un campo di testo standard. Ne consegue che scrivere `<input>, <input type="fattiforza">` oppure `<input type="text">` produce sempre un campo di testo.

Questo accade in omaggio al principio per cui ogni browser deve ignorare i tag che non riconosce e gli attributi che non sa interpretare anche in relazione a elementi che gli sono noti. Ecco perché già oggi è possibile trarre vantaggio dalle innovazioni proposte, dando modo ai browser più evoluti di offrire funzionalità avanzate, senza essere d'impaccio a chi naviga usando browser più datati o magari, anche se recenti, meno rispettosi della specifica; in ogni caso l'utente otterrà un servizio disponibile e funzionante.

## search

La specifica prevede espressamente che questo campo sia visualizzato in modo da essere conforme alle caratteristiche di stile della piattaforma in uso. In altri termini, potrà essere prevista una diversa modalità di visualizzazione per i campi di ricerca. A prima vista sembra poca cosa, ma utilizzando questo controllo si ottengono immediatamente almeno due vantaggi, che possono essere riassunti in: migliore accessibilità e minore sforzo cognitivo.

### Migliore accessibilità

Le tecnologie assistive (si pensi agli screen reader come Jaws che consentono a non vedenti e agli ipo-vedenti di navigare sul Web) sono messe nelle condizioni ideali per identificare in modo univoco il campo di ricerca e decidere, per esempio, di assegnare a questo campo il focus, indipendentemente dalla collocazione dello stesso nella pagina web.

### Navigare al "buio"

Mai sentito parlare di "Dialogo nel Buio"? È una mostra ospitata nella sede dell'Istituto dei ciechi a Milano. Si tratta di un percorso istruttivo perché, per pochi minuti, simula la vita di un non vedente in vari contesti. Altrettanto istruttivo è fare un viaggio nel mondo dei bit con Jaws (o un qualunque altro software che abbia funzioni analoghe): usare uno screen reader per navigare sul Web dà una misura, spesso impietosa, del grado di usabilità di un sito.

### Minore sforzo cognitivo

Lo sforzo cognitivo può essere definito, con una certa approssimazione, come lo sforzo necessario per elaborare una data informazione. Marcare il campo di ricerca con un `<input type="search">` lascia al browser la possibilità di attribuirgli un'identità distinta rispetto ad altri controlli, il che ne consente una rapida identificazione all'interno della pagina web. Safari e Chrome offrono un esempio eclatante.

Safari su Mac mostra un campo di testo con angoli arrotondati (Figura 3.1), diversamente da quanto accade con altri controlli.



**FIGURA 3.1** Campo di ricerca con angoli arrotondati, senza focus.

Non appena si iniziano a digitare caratteri all'interno dell'elemento, nella parte sinistra dello stesso appare una piccola "x" (Figura 3.2).



**FIGURA 3.2** Nella parte sinistra del campo una piccola "x" funge da pulsante di reset.

Un clic su tale piccolo pulsante e il testo inserito scompare! Per l'utente che ha familiarità con il Mac il funzionamento del campo di ricerca risulta immediato, perché l'aspetto stilistico e le modalità di interazione sono identiche a quelle cui è già abituato. Lo sforzo cognitivo dell'utente è pari a zero perché l'utente interagisce con elementi noti!

In Chrome, indipendentemente dal sistema operativo in uso, l'elemento appare del tutto identico a un campo di testo, ma anche in questo caso, quando si inizia a inserire del testo, appare l'icona di una "x" che, se selezionata con un clic, consente di ripulire il campo (Figura 3.3).



**FIGURA 3.3** L'icona di una "x" segnala la possibilità di reimpostare il testo di ricerca.

Tutti gli altri browser, sino a oggi, non prevedono una visualizzazione diversa da quella di un ordinario campo di testo.

## email

Una delle informazioni che capita più spesso di inserire in un form è l'indirizzo di posta elettronica. In fondo è anche per questa assillante richiesta che sono nati gli indirizzi di posta elettronica temporanei.

## Mailbox temporanee

Tutto è iniziato con "10 minute mail". Si tratta di un servizio gratuito che consente di creare un indirizzo di posta per il tempo necessario a provvedere all'odiata registrazione su quei siti che si è certi di non voler utilizzare più di una volta o per cui non si tollera ricevere anche un solo byte di spam. Il successo di 10 minute mail è stato sorprendente. Da allora, altri servizi analoghi si sono diffusi rapidamente in Rete: basta fare una ricerca su Google con le parole chiave "temporary email" per farsi un'idea del panorama attuale.

HTML5 introduce un controllo specifico per l'email che, con slancio di iperbolica fantasia, richiede il valore... `email`. Ecco il codice di un possibile form di login:

### **LISTATO 3.2** Form di login con input type "email"

```
<form method="post" action="http://html5.gigliotti.it/process_data.php">
  <fieldset>
    <legend>Accedi</legend>
    <p>
      <label for="email" required>Email</label>
      <input type="email" id="email" name="email" size="30">
    </p>
    <p>
      <label for="password">Password</label>
      <input type="password" id="password" name="password" size="30">
    </p>
```

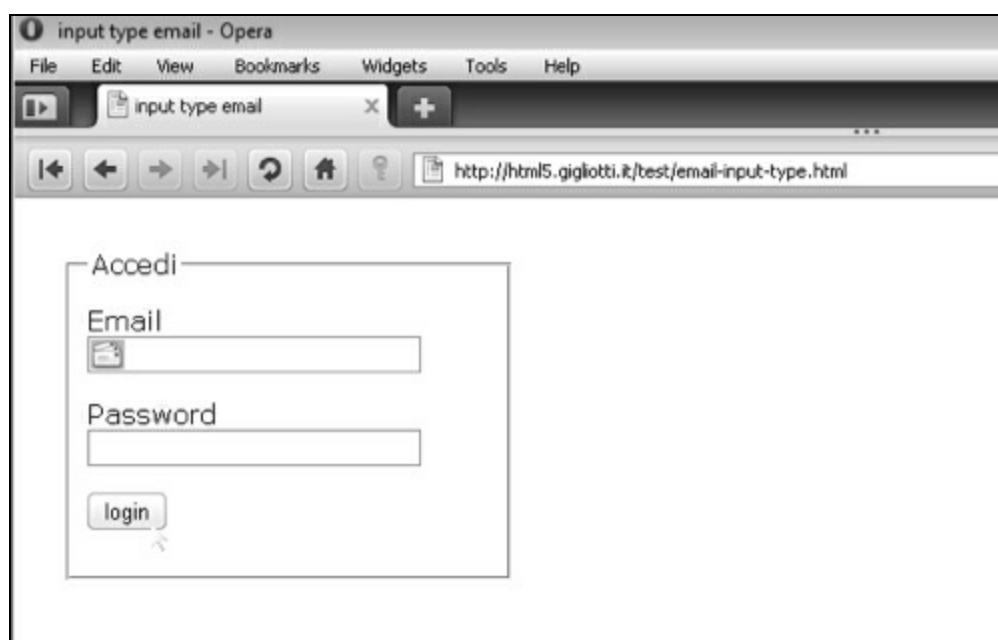
```

</p>
<p>
  <input type="submit" name="login" value="login">
</p>
</fieldset>
</form>

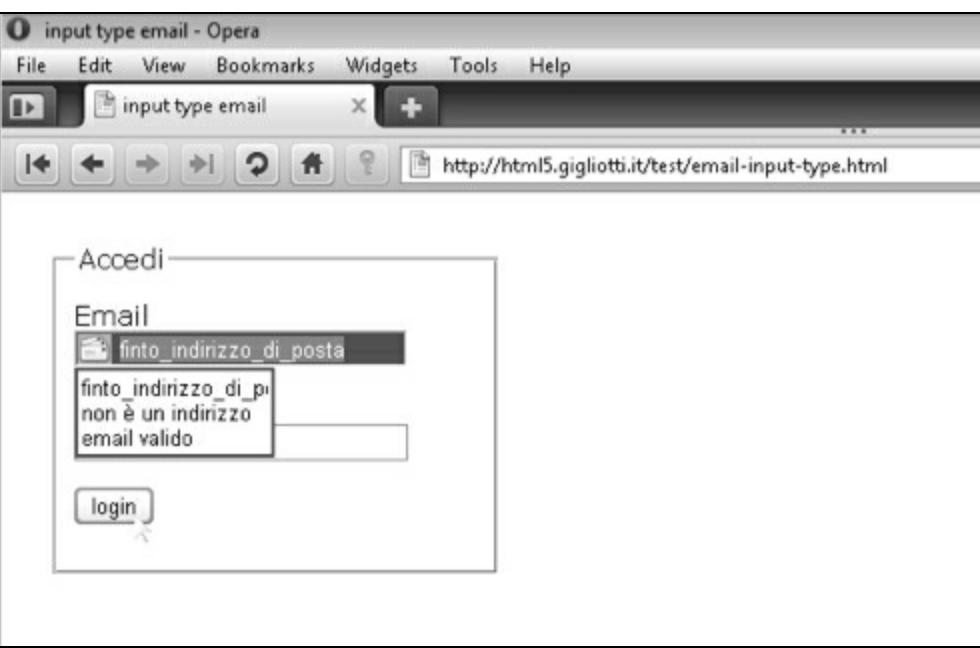
```

Il tipo `email` è rappresentato graficamente, nella quasi totalità dei casi, come un ordinario campo di testo. Attualmente fa eccezione solo il browser Opera, che introduce un elemento stilistico: un'icona rappresentante 3 lettere affrancate che appare all'inizio del campo Email (Figura 3.4).

Opera utilizza in modo efficace il nuovo elemento offrendo un controllo di conformità dell'indirizzo di posta elettronica. Infatti, se si prova a inoltrare il form di autenticazione fornendo un indirizzo fasullo, per esempio `finto_indirizzo_di_posta`, al momento di inoltrare i dati premendo il pulsante login il browser avvisa l'utente che l'indirizzo email fornito non è valido (Figura 3.5).



**FIGURA 3.4** Opera caratterizza il campo Email con l'icona di una busta da lettera (ma non su Linux!).



**FIGURA 3.5** Opera avvisa l'utente che l'indirizzo inserito non risulta essere valido.

## Lo stile nella validazione: lavori in corso

Dal punto di vista grafico, il box bianco con bordo rosso ha sollevato tra gli addetti ai lavori qualche perplessità sia per la poco piacevole rappresentazione grafica, sia per l'impossibilità di intervenire attraverso fogli di stile per migliorarne l'aspetto o anche solo integrarne la modalità di visualizzazione con altri elementi grafici del sito. Il team di sviluppatori di Opera ha iniziato a lavorare a queste segnalazioni per produrre una grafica più accattivante: sembra che il messaggio di errore verrà proposto in una nuvoletta simile a quelle che appaiono nei fumetti. Inoltre, è probabile siano riformulati anche i messaggi di errore.

Solo se si è provato a implementare un controllo di conformità dell'indirizzo di posta elettronica si può apprezzare appieno questo supporto che il browser offre allo sviluppatore e all'utente a costo zero.

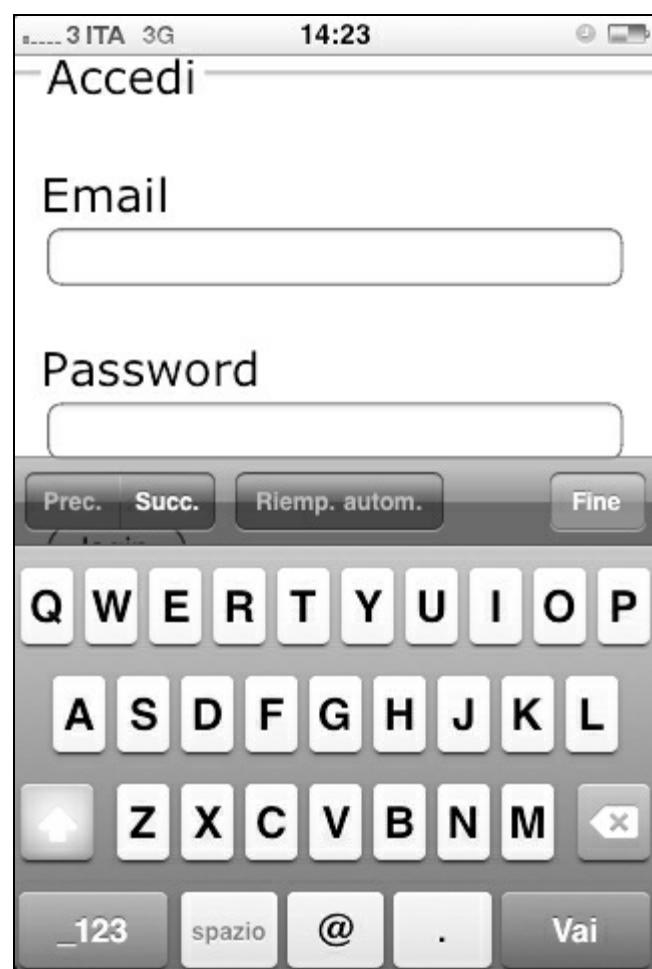
## Le difficoltà di validazione di un indirizzo email

La validazione di un indirizzo email non è banale da implementare nemmeno ricorrendo alle potenti espressioni regolari, che anche HTML5 mette a disposizione mediante il nuovo attributo `pattern`, di cui tratteremo più avanti. Su questo tema la specifica introduce una "violazione intenzionale" rispetto alla specifica RFC 5322 che detta la sintassi relativa agli indirizzi di posta elettronica. Ciò accade perché, cito testualmente quanto riportato a riguardo sul W3C (<http://www.w3.org/TR/html5/states-of-the-type-attribute.html>), "la specifica (RFC 5322) è contemporaneamente troppo rigorosa (prima della chiocciola), troppo vaga (dopo la chiocciola), e troppo accomodante (per via del fatto che acconsente l'uso di commenti, spazi, e doppi apici, il tutto in modo poco familiare alla maggior parte degli utenti) per essere di uso pratico".

Anche Chrome 5 adotta un controllo dell'indirizzo di posta, ma è più criptico di Opera: si

limita infatti a inibire l'inoltro del form portando il focus sul campo contenente l'indirizzo email non conforme.

Fatta eccezione per i browser sopra citati, non si noterà alcuna differenza tra un nuovo campo email e un ordinario campo di testo. Tuttavia, se si considera il mondo mobile, scopriamo che l'iPhone, con il suo browser predefinito Safari Mobile, nel momento in cui il campo email assume il focus assiste l'utente con una speciale tastiera touch che risulta diversa da una "tastiera tradizionale" (la QWERTY), come mostrato in Figura 3.6.



**FIGURA 3.6** La tastiera touch proposta da Safari Mobile su iPhone dedicata al campo email.

## La tastiera QWERTY

Milwaukee, nel Wisconsin, non è solo la città dove sono stati ambientati gli episodi di Happy Days: è anche il luogo dove, nel lontano 1873, Christopher Latham Sholes ideò la disposizione dei tasti sulla tastiera come li conosciamo oggi. Tale disposizione prende il nome di QWERTY, dalla sequenza dei 6 caratteri posti più in alto a sinistra.

La tastiera che abbiamo davanti tiene conto delle limitazioni cui deve sottostare un indirizzo di posta elettronica e presenta tasti speciali che semplificano l'inserimento del dato richiesto. Un tasto per la chiocciola e un tasto per il punto trovano posto di fianco a una barra spaziatrice di dimensioni ridotte.

Disporre di un campo di testo specifico per gli indirizzi di posta elettronica spalanca le

porte a una serie di tecnologie di supporto. L'iPhone sfrutta la maggiore precisione sul tipo di dato richiesto (un indirizzo di posta elettronica, appunto) per offrire all'utente un'interfaccia più amichevole.

Questo controllo accetta il nuovo attributo `multiple`. Si tratta di un attributo booleano che, se specificato, prevede l'inserimento di una lista di indirizzi email validi separati da virgola. Un tipico esempio di campo in cui si inseriscono più indirizzi email è il campo CC di un form di invio email (come Google Gmail o Hotmail).

### **LISTATO 3.3** Esempio d'uso per l'attributo `multiple`

```
<label for="cc">Cc:</label> <input type="email" id="cc" name="cc" multiple>
```

## Attributi booleani in HTML

Un elemento si dice booleano quando può assumere solo 2 valori: "true" o "false", "0" o "1" ecc. In HTML esistono due notazioni per definire il valore di un attributo booleano. La prima è la sintassi breve, secondo la quale la sola presenza dell'attributo corrisponde al caso "true", mentre se l'attributo è omesso si ha il caso "false". Si veda l'esempio sopra riportato per l'attributo `multiple`: è stato sufficiente indicarne il nome per impostarne a "true" il suo valore.

Qualora invece all'attributo booleano sia attribuito un valore, questo deve essere pari al suo stesso nome. Sempre per rifarsi all'esempio dell'attributo `multiple`, si avrà:

### **LISTATO 3.4** Esempio d'uso per l'attributo `multiple`, sintassi estesa

```
<label for="cc">Cc:</label> <input type="email" id="cc" name="cc" multiple="multiple">
```

Quest'ultima è la sintassi estesa.

## url

Un altro dato ricorrente nei form è quello relativo all'indirizzo web. Una delle informazioni che i social network come Twitter, Facebook, LinkedIn chiedono di inserire per completare il proprio profilo è l'indirizzo del blog o del sito hobbistico/professionale. Definire un campo `url` è semplice:

### **LISTATO 3.5** Form con input type "url"

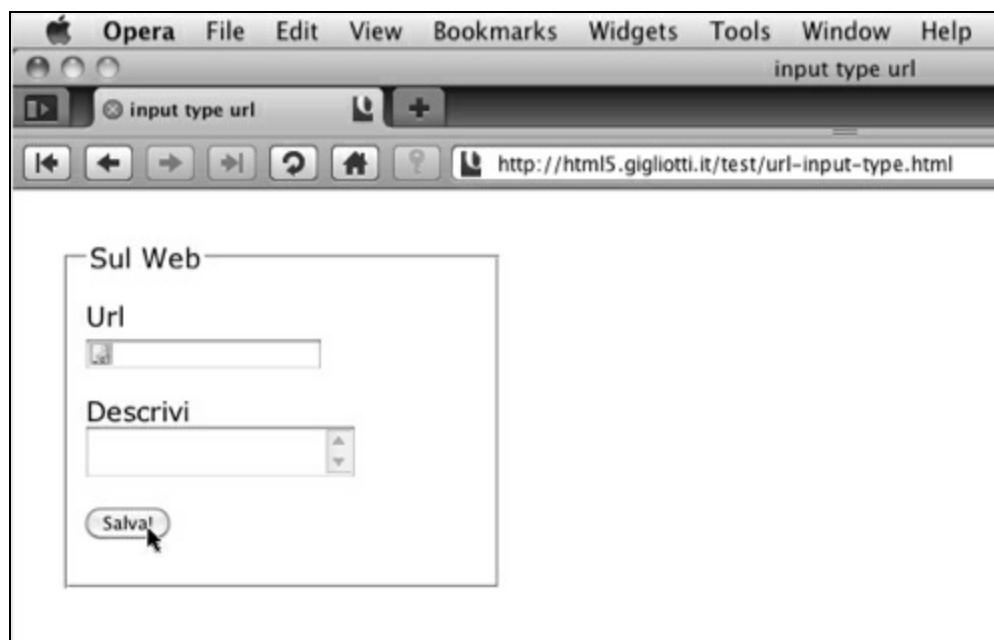
```
<form method="post" action="http://html5.gigliotti.it/process_data.php">
<fieldset>
<legend>Sul Web</legend>
```

```

<p>
  <label for="url_address" required>Web</label>
  <input type="url" id="url_address" name="url_address">
</p>
<p>
  <label for="web_desc" required>Descrivi</label>
  <textarea id="web_desc" name="web_desc"></textarea>
</p>
<p>
  <input type="submit" name="salva" value="Salva!">
</p>
</fieldset>
</form>

```

Il layout non ci consente di percepire alcuna differenza rispetto a un comune campo di testo. Anche in questo caso è sempre il browser Opera a distinguersi con l'aggiunta di un'icona raffigurante un documento sullo sfondo di una stella, forse non proprio una scelta felice per identificare un indirizzo web (Figura 3.7).



**FIGURA 3.7** Caratterizzazione stilistica del campo `<input type="url">` proposta da Opera.

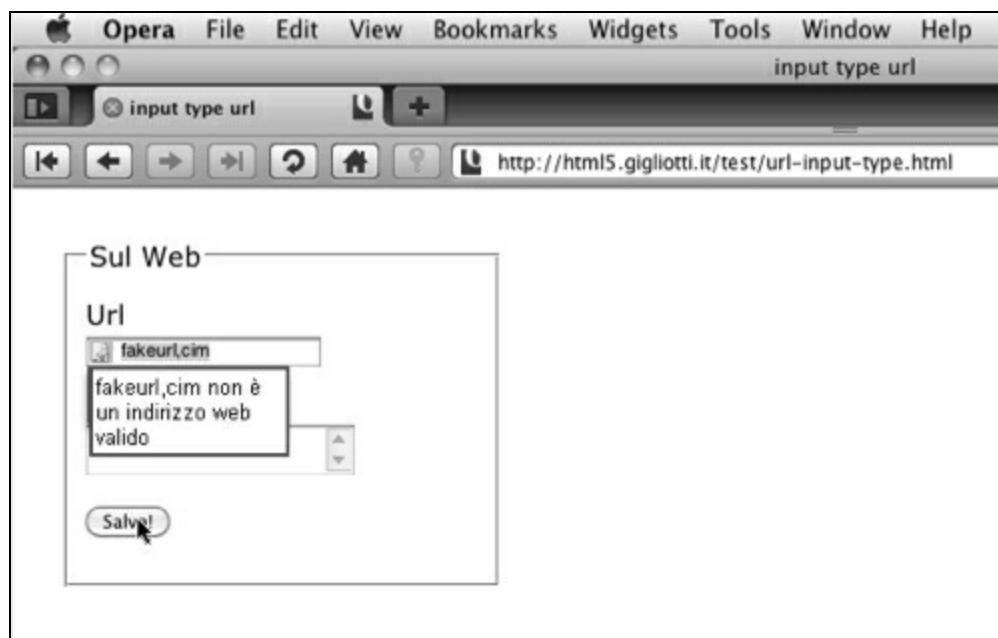
Più importante della caratterizzazione stilistica è il controllo di conformità che non solo Opera, ma anche Chrome, rende disponibile quando si tenta di inoltrare un indirizzo web non conforme (Figura 3.8).

Safari Mobile, su iPhone, sfrutta questa informazione specifica fornendo una tastiera touch dedicata (Figura 3.9).

Non è presente la barra spaziatrice, al suo posto appaiono tre pulsanti dedicati: uno per il punto, uno per lo slash e uno per il dominio `.com` (altri domini sono a portata di mano tenendo premuto a lungo questo pulsante).

**LISTATO 3.6** Form con input type "tel"

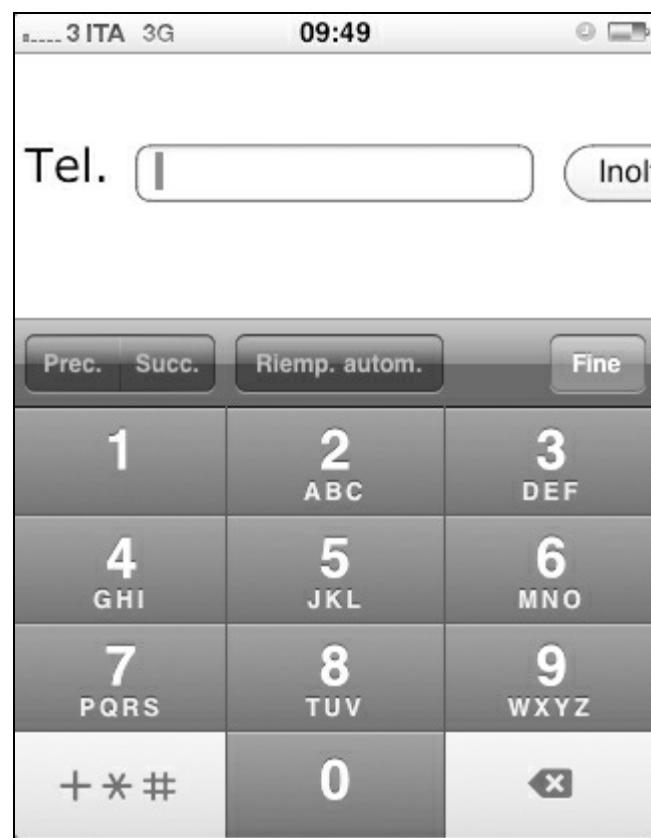
```
<form method="post" action="http://html5.gigliotti.it/process_data.php">  
<ul>  
<li>  
  <label for="telefono">Tel.</label>  
  <input type="tel" id="telefono" name="telefono">  
  <input type="submit" name="inoltra" value="Inoltra">  
</li>  
</ul>  
</form>
```

**FIGURA 3.8** Opera segnala che l'indirizzo immesso non è valido.



**FIGURA 3.9** Tastiera touch dedicata per il campo `<input type="url">`.

Ancora una volta, ecco un campo che non presenta differenze, in termini di layout, rispetto a un ordinario campo di testo. Scopriamo il valore aggiunto, oltre alla migliore definizione semantica, sempre grazie all'iPhone e alle sue piccole tastiere touch dedicate (Figura 3.10).



**FIGURA 3.10** Tastiera touch dedicata per il campo `<input type="tel">`.

Anche in questo caso ne viene proposta una che risponde alle specifiche esigenze di chi deve digitare un numero di telefono. È probabile che in futuro si possa raggiungere qualche interazione con la propria rubrica in modo da facilitare ulteriormente l'inserimento di questo dato. Tuttavia appare difficile si possa introdurre un tipo di validazione efficace, a causa delle grandi differenze esistenti tra i numeri di telefono internazionali.

## number

`<input type="number">` è un controllo dedicato all'inserimento di valori numerici. Dal punto di vista stilistico si presenta come un campo di testo arricchito da due piccole frecce poste nella parte sinistra del controllo. Una freccia è rivolta verso l'alto e serve per incrementare il valore; l'altra è rivolta verso il basso e agisce decrementando il valore inserito. Laddove questo elemento è supportato cambiano alcuni dettagli di rappresentazione. Con Opera le due frecce appaiono al di fuori di un piccolo campo di testo come due pulsanti separati, mentre Chrome disegna le frecce all'interno del campo, e il campo di testo è decisamente più lungo rispetto alla visualizzazione proposta da Opera (Figura 3.11).



Una volta raggiunto il valore massimo o minimo, dunque una volta arrivata a "fine corsa", la freccia corrispondente diventa grigia e non più attiva. Un controllo numerico con le caratteristiche qui descritte prende il nome di spinbox. A esso si applicano i nuovi attributi `max`, `min`, `step`:

- `min` definisce il valore minimo dell'intervallo e, se non viene specificato, come nell'esempio sopra, assume il suo valore predefinito pari a zero;
- `max` definisce il valore massimo dell'intervallo e, se non viene specificato, il suo valore predefinito è 100;
- `step` regola il passaggio da un valore al successivo nell'intervallo scelto.

`value` è un vecchio attributo che, in questo contesto, permette di impostare il numero predefinito.

Si può comprendere meglio l'utilità di questo nuovo strumento con un esempio.

#### **LISTATO 3.7** I molteplici usi del controllo number

```
<form method="post" action="http://html5.gigliotti.it/process_data.php">
  <fieldset>
    <legend>Diamo i numeri!</legend>
    <p>
      <label for="num_libri">Libri letti: </label>
      <input type="number" id="num_libri" name="num_libri" min="0" max="20" step="1" value="1">
      <label for="num_gradi">Palline da golf <span>(in conf. da 5)</span>:</label>
      <input type="number" id="num_golf" name="num_golf" min="0" step="5" value="5">
      <label for="num_gradi">Temperatura minima: </label>
      <input type="number" id="num_gradi" name="num_gradi" min="-50" max="30" step="1" value="-10">
      <label for="num_kg">Peso <span>(in Kg)</span>:</label>
      <input type="number" id="num_kg" name="num_kg" min="0" max="140" step="0.1" value="75.5">
      <input type="submit" id="inoltra" name="inoltra" value="Inoltra">
    </p>
  </form>
```

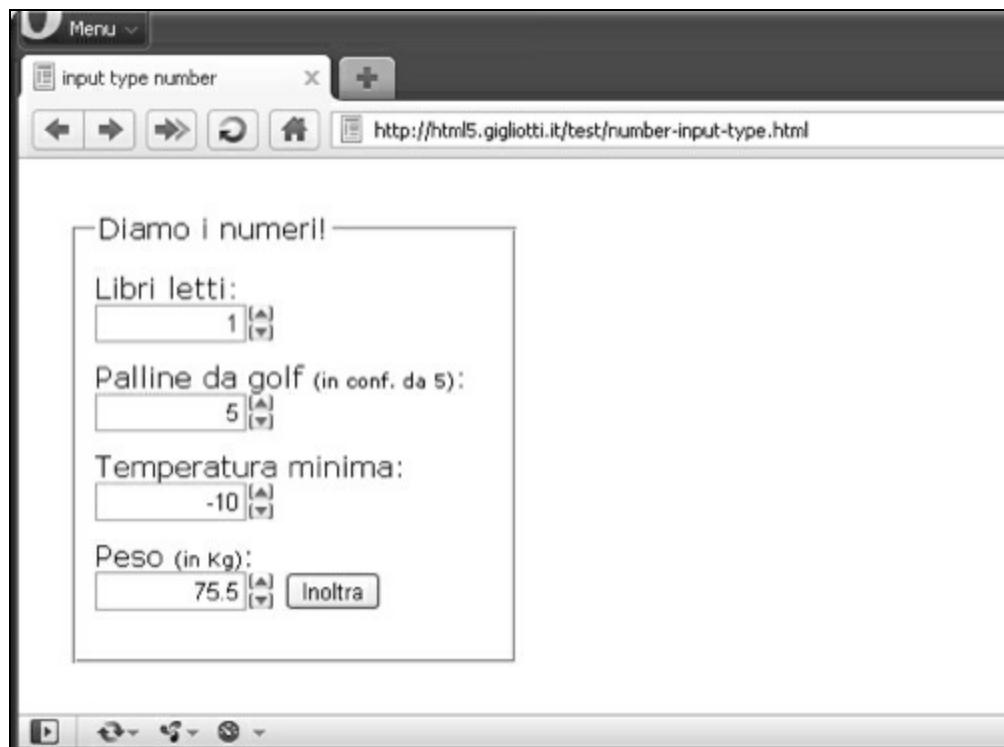
In Opera questo listato produce una pagina web come quella rappresentata in Figura 3.12.

Analizziamo uno a uno gli spinbox del listato. Possiamo toccare con mano la versatilità di questo controllo e il modo in cui si presta come valido supporto alla selezione del valore numerico.

- Libri letti. Il primo controllo invita l'utente a inserire un numero che, auguriamoci

per l'utente stesso, è un intero positivo: un tocco di sano realismo spinge a includere anche lo zero in questo intervallo. Impostando valore minimo (`min="0"`), massimo (`max="20"`), un fattore di incremento/decremento (`step="1"`), così come un valore predefinito da mostrare all'utente prima che questi abbia modo di interagire con il controllo (`value="0"`), si determinano una serie di valori considerati ammissibili all'interno dell'intervallo predefinito. I valori ammissibili sono tutti i numeri interi positivi da 0 a 20, estremi inclusi (0, 1, 2, 3 ecc.).

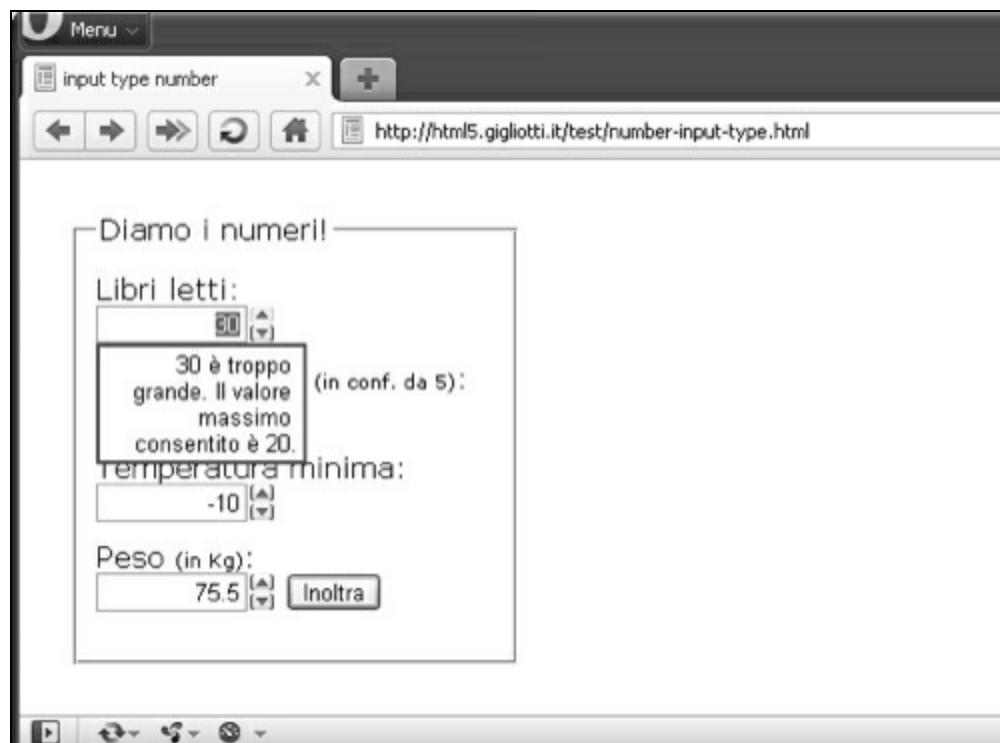
- Palline da golf (in conf. da 5). In questo contesto i soli numeri accettabili sono 0, 5 o multipli di 5. Impostando un valore minimo pari a 0 (`min="0"`) e un fattore di variazione pari a 5 (`step="5"`) si restringe il campo dei valori ammissibili a quelli effettivamente richiesti.
- Temperatura minima. Questo terzo controllo consente di specificare numeri interi negativi. Il fattore di variazione è pari a 1.
- Peso (in kg). In ultimo si definisce un intervallo di numeri reali, numeri con una parte decimale (il separatore della parte decimale è un punto “.” e non la virgola; ciò vale anche nella versione del browser in lingua italiana). Il fattore di incremento/decremento consente di applicare variazioni di 100 grammi, ossia 0,1 kg.



**FIGURA 3.12** Campi spinbox per l'inserimento di valori numerici.

È sempre possibile inserire in uno spinbox valori diversi da quelli selezionabili attraverso le sue frecce, tuttavia i browser che implementano questo controllo segnalano l'incongruenza tra il valore inserito e quelli dichiarati come validi. Opera mostra un messaggio in box bianco con bordo rosso (Figura 3.13).

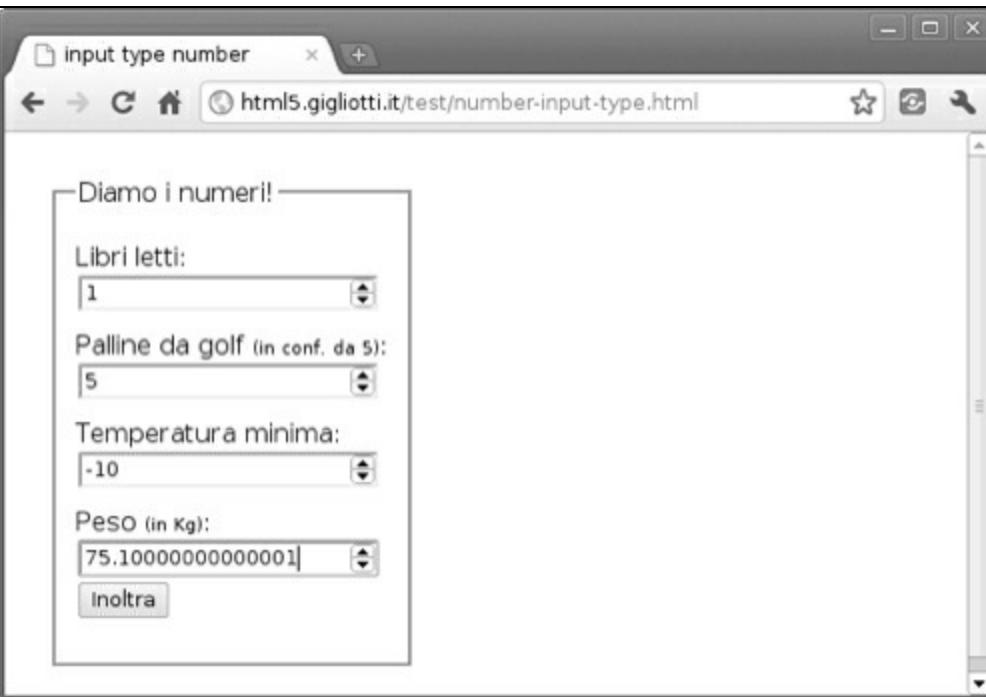
Chrome, invece, come accade per i controlli analizzati in precedenza, inibisce l'invio e restituisce il focus al campo che ha violato il set di valori dichiarati come accettabili, senza l'aggiunta di alcun messaggio di avvertimento.



**FIGURA 3.13** Opera inibisce l'invio del form se il valore che si propone non soddisfa i vincoli posti dal controllo.

## Google Chrome e gli arrotondamenti

Se il valore assegnato all'attributo `step` non è un numero intero (come nel caso dello spinbox utilizzato per inserire il peso in kg, dove `step="0.1"`), può succedere che Chrome commetta un errore nel calcolare il valore proposto all'utente (Figura 3.14).



**FIGURA 3.14** Chrome soffre di un errore infinitesimale nel calcolo dei valori decimali.

## range

La specifica stabilisce che si ricorre a questo controllo quando si vuole impostare un valore numerico con l'esplicito ammonimento che il valore esatto non sia da considerarsi importante. `<input type="range">` produce un'interfaccia che prende il nome di **slider**. Il controllo è composto da due elementi: una barra e un puntatore. Quest'ultimo viene fatto scorrere lungo la barra per segnalare la scelta di uno dei valori selezionabili lungo la scala.

Ecco un esempio:

### **LISTATO 3.8** Uno slider HTML5

```
<label for="range">Indice di gradimento</label>
<input type="range" id="range" name="range">
```

Ci sono differenze di stile tra Opera, Chrome e Safari, differenze accentuate dal sistema operativo su cui questi browser sono installati. Le tre immagini che seguono (Figure 3.15, 3.16 e 3.17) mettono in mostra lo slider generato da Safari 5 su Mac e su Windows (XP/Win7), a ulteriore conferma di quanto il sistema operativo giochi un ruolo importante nella definizione del layout di questi elementi.



**FIGURA 3.15** Lo slider proposto da Safari 5 su Mac 10.6.



**FIGURA 3.16** Lo slider proposto da Safari 5 su XP SP2.



**FIGURA 3.17** Lo slider proposto da Safari 5 su Windows 7.

A quanto pare, i browser che fin qui hanno implementato il controllo hanno preso alla lettera quella parte della specifica in cui si afferma che, con l'adozione di questo elemento, il valore esatto non è importante! E non potrebbe essere altrimenti poiché, anche impostando gli attributi `min`, `max` e `step`, si continua a non avere alcun riferimento numerico lungo lo slider che possa far comprendere meglio quale valore si è posto in input! Per migliorare l'usabilità del controllo si può fare ricorso al tag `<output>`.

Anche questo controllo, come tutti gli altri elencati finora, se non implementato dal browser sarà visualizzato come un ordinario campo di testo.

### ATTENZIONE

Lo slider impone sempre di impostare un valore. Mentre un campo di testo, se non obbligatorio, potrà anche essere inoltrato "in bianco", questo non accade mai per lo slider, perché l'indicatore verrà sempre posto su uno dei possibili valori dell'intervallo prescelto.

L'utente si troverà sempre di fronte a un form perfettamente funzionante con cui interagire, ma l'usabilità dello strumento ne risulterà compromessa. Questo è uno dei casi in cui è bene adottare una strategia che garantisca un'alternativa per quegli utenti che non dispongono di un browser che rispetta le specifiche HTML5.

## Strategia alternativa

Il primo passo da compiere è quello di identificare il supporto da parte del browser dell'utente: solo così possiamo stabilire se il controllo nativo HTML5 sia sufficiente o se, viceversa, si debba ricorrere a una soluzione alternativa, quale per esempio il controllo di tipo slider proposto dalla libreria JavaScript `jQuery UI`.

Ci affidiamo a `Modernizr` per stabilire se il browser implementa o meno il controllo che intendiamo visualizzare.

## **LISTATO 3.9** Inclusione della libreria Modernizr

```
<script src="modernizr-1.6.min.js"></script>
```

L'ultima versione della libreria è disponibile all'indirizzo <http://www.modernizr.com>. Notiamo subito una particolarità dell'elemento `<script>`: di solito siamo abituati a vedere una sintassi più verbosa, come quella del Listato 3.10.

## **LISTATO 3.10** Sintassi del marcatore `<script>`

```
<script src="modernizr-1.6.min.js" language="JavaScript" type="text/javascript"></script>
```

Tuttavia l'attributo `language` è stato deprecato perché mai riconosciuto come standard. Per definire il linguaggio di scripting adottato si utilizza invece l'attributo `type`. In HTML5 questo attributo è stato reso opzionale se tale linguaggio è JavaScript. In considerazione del fatto che in questo esempio e in quelli riportati nei prossimi capitoli si fa sistematicamente ricorso a JavaScript, si utilizza sempre il marcatore `<script>` nella forma più snella illustrata nel Listato 3.9.

Una volta inclusa la libreria possiamo concentrarci sul test di supporto vero e proprio.

## **LISTATO 3.11** Test di supporto dell'attributo range con Modernizr

```
<script>
if (Modernizr.inputtypes.range) {
  // supporto HTML5 nativo per <input type="range">
} else {
  // nessun supporto. Si ricorre a libreria JavaScript
  // ad es. jQuery per proporre uno slider alternativo.
}
</script>
```

Modernizr utilizza una mappa chiamata `inputtypes` per cui a ognuna delle 13 chiavi definite (`search, tel, url, email, datetime, date, month, week, time, datetime-local, number, range, color`) corrisponde il valore booleano `true` o `false` relativo al supporto di quel dato elemento. Se `Modernizr.inputtypes.range` restituisce `true`, il controllo slider rappresentato da `<input type="range">` è supportato dal browser. Se invece questa condizione non è soddisfatta (`Modernizr.inputtypes.range = false`), si dovrà ricorrere a una libreria esterna, per esempio jQuery, per proporre quel tipo di controllo.

Riferimenti alle risorse citate e altri link utili

- La specifica relativa ai form: <http://www.w3.org/TR/html5/forms.html>
- Jaws è uno screen reader, letteralmente un lettore di schermo. Il software è realizzato da Freedom Scientific: <http://www.freedomscientific.com/jaws-hq.asp>
- Il sito web della mostra “Dialogo nel Buio”: <http://www.dialogonelbuio.org>
- “10 Minute Mail” è l’antesignano dei servizi di posta elettronica temporanea: <http://10minutemail.com>
- L’articolo pubblicato sul blog ufficiale del browser Opera in cui viene illustrata un’anteprima, fruibile per mezzo del video proposto a corredo, del nuovo stile di rappresentazione dei messaggi di errore: <http://my.opera.com/ODIN/blog/html5-forms-error-reporting-with-wobbly-bubbles>

## Conclusioni

Proseguirà nel prossimo capitolo la lettura delle novità introdotte con HTML5 su questo argomento. Da quanto sin qui documentato emerge un’implementazione delle principali funzionalità ancora frammentata. Questo può rendere necessario il ricorso a strategie alternative in tutti quei casi in cui il mancato riconoscimento dei nuovi marcatori e attributi può compromettere l’esperienza d’uso dei nostri visitatori.

# Web Forms 2.0 (seconda parte)

In questo capitolo prosegue l'esplorazione dei nuovi controlli che HTML5 mette a disposizione per la creazione di form evoluti. Analizzeremo i controlli di calendario che semplificano l'inserimento di data e ora, il color picker (uno strumento di gestione dei colori) e il datalist. Ci si soffermerà su alcuni attributi che migliorano sensibilmente l'interfaccia utente (`autocomplete`, `autofocus`, `form`, `pattern`, `placeholder`, `required`). Tutti questi aggiornamenti rendono i form uno strumento potente: "Da grandi poteri derivano grandi responsabilità", era il monito di zio Ben al giovane Peter Parker, ma se il vostro "senso di ragno" non vi fosse d'aiuto, troverete alcuni suggerimenti nell'ultimo paragrafo di questo capitolo.)



**FIGURA 4.1** Se il "senso di ragno" non ci aiuta, passiamo in rassegna la lista dei suggerimenti in coda al capitolo [Uomo Ragno, "Ciudad de las Artes y las Ciencias", Valencia, Spagna]

## Controlli di calendario

Fino alla nascita di HTML5 non esistevano controlli che consentissero di impostare in modo semplice informazioni come data e ora. A questa mancanza si sopperisce oggi con sei nuovi controlli dedicati allo scopo: `date`, `datetime`, `datetime-local`, `month`, `week`, `time`. A tutti i controlli di calendario si applicano gli stessi attributi `min`, `max` e `step` introdotti nel capitolo precedente per i controlli numerici.

# date

I date picker sono quei piccoli calendari creati attorno a campi di testo per rendere più agevole la scelta e l'inserimento automatico della data. Si tratta di strumenti ormai molto diffusi. Qualunque sito di prenotazioni online (che si tratti di biglietti aerei, hotel, o auto a noleggio non fa differenza) fa ampio uso di questo modello. Per quanto siano diffusi non esiste, tuttavia, alcuno standard per la loro rappresentazione: l'unica certezza è che non si troverà mai un controllo di un calendario che sia identico a un altro. In un caso la visualizzazione del calendario è scatenata dal focus sul campo di testo, in un altro caso è necessario fare clic sull'icona, sempre diversa, che rappresenta il calendario e che normalmente è posta al lato del campo in cui verrà poi riportata la data prescelta. Con l'ausilio di un controllo standard anche gli utenti che hanno meno dimestichezza con il Web hanno poche difficoltà di interazione, trattandosi di una riduzione dello sforzo cognitivo.

## LISTATO 4.1 Impostare una data

```
<label for="datepicker">Scegli una data: </label>
<input type="date" id="datepicker" name="datepicker">
```

Tutti gli attributi già introdotti per i controlli numerici sono validi anche per i controlli di calendario.



**FIGURA 4.2** Il date picker standard com'è proposto da Opera.

## min, max e step: la chiave di lettura è data dal contesto

Il valore predefinito dell'attributo `step` cambia di controllo in controllo (per esempio questo valore è 1 per il campo `date` ma 60 per il campo `time`). Non solo, anche se il valore predefinito è lo stesso, il suo significato cambia in base al controllo cui è attribuito. Per esempio: scrivere `step="1"` significa introdurre un "salto" di 1 settimana se l'attributo si riferisce al controllo `week`, mentre lo stesso valore implica un "salto" di un mese se il controllo si riferisce a un controllo di tipo `month`.

Se l'attributo `step` non è stato impostato, il suo valore predefinito è pari a 1 (giorno). Si interviene sulle date disponibili per la selezione avendo cura di scegliere una data di inizio e una di fine, rispettivamente attraverso gli attributi `min` e `max`. Agendo su `step` si interviene sull'insieme di dati disponibili nell'intervallo: giocare con questi attributi è come esercitarsi con l'insiemistica.

#### **LISTATO 4.2** La selezione della data è limitata mediante l'uso di `min`, `max` e `step`

```
<label for="datepicker">Scegli una data: </label>  
<input type="date" id="datepicker" name="datepicker" step="2" min="2010-11-01" max="2010-11-30">
```



**FIGURA 4.3** È possibile limitare in modo altamente personalizzabile il numero di opzioni oggetto di scelta.

In questo modo il calendario restringe la possibilità di scelta ai soli giorni dispari del mese di novembre, come illustrato in Figura 4.3.

## time

Il tipo `time` è rappresentato tanto su Opera quanto su Chrome come uno spinbox (per una definizione di spinbox si veda il capitolo precedente), concepito appositamente per l'inserimento di ora, minuti e, in via facoltativa, anche secondi.



**FIGURA 4.4** Lo spinbox per l'impostazione dell'ora.

#### **LISTATO 4.3** Impostare un orario

```
<label for="timepicker">Ora: </label>  
<input type="time" id="timepicker" name="timepicker" value="00:00">
```

L'attributo `step` non è presente nel listato HTML, dove viene usato il suo valore predefinito che è pari a 60 (secondi).

Se per questo attributo si opta per un valore diverso da 60 o da un suo multiplo, la visualizzazione del controllo si arricchisce della nuova informazione relativa ai secondi, che è ora possibile impostare (Figura 4.5).



**FIGURA 4.5** Diventa possibile impostare anche i secondi quando si sceglie un valore diverso da 60 o un suo multiplo per l'attributo `step`.

Anche per questo controllo si interviene sui valori disponibili per intervallo impostando `min`, `max` e `step`.

Ecco un controllo che consente di selezionare le ore tra le 10 e le 12 con intervalli di 15 minuti. L'attributo `step` in questo caso accetta solo definizioni in termini di secondi, quindi si imposta `step` pari a 900, ossia il numero di secondi presenti in 15 minuti.

**LISTATO 4.4** Impostare un orario: dalle 10:00 alle 12:00 con intervalli di 15 minuti

```
<label for="timepicker">Ora: </label>
<input type="time" id="timepicker" name="timepicker" value="10:00" min="10:00" max="12:00" step="900">
```

## datetime e datetime-local

Possiamo considerare sia `datetime` sia `datetime-local` come la somma dei due controlli precedenti; in effetti ciascuno dei due accosta al calendario, utile per impostare giorno, mese e anno, uno spinbox da utilizzare per l'indicazione delle ore, dei minuti e dei secondi. La differenza tra i due controlli sta nell'informazione sul fuso orario, presente in `datetime` e assente in `datetime-local`. Quest'ultimo definisce un'informazione data/ora riferita a un contesto locale, ossia senza alcuna nozione relativa al fuso orario di appartenenza. La presenza o l'assenza di questa informazione si riflette anche sul layout dell'elemento.

Il valore predefinito dell'attributo `step` qui è pari a 60 (secondi).

**LISTATO 4.5** Impostazione di data e ora (con informazioni relative al fuso orario)

```
<label for="datetimepicker">Imposta data e ora: </label>
<input type="datetime" id="datetimepicker" name="datetimepicker">
```



**FIGURA 4.6** Impostazione di data e ora con informazioni sul fuso orario (UTC).

min e max, in questo contesto, vanno popolate secondo quanto riportato in Figura 2.2 (si veda il Capitolo 2). Il formato ISO della data, decomposto nei suoi elementi base, consente di restringere la possibile selezione di data/ora a un intervallo ben definito. Nel Listato 4.6 sono riportati un paio di esempi.

**LISTATO 4.6** Alcuni esempi di applicazione degli attributi min e max al controllo datetime

```
<input type="datetime" min="2010-11-01T13:20Z" max="2010-11-15T10:30Z">
<input type="datetime" min="2010-11-01T13:20+01:00" max="2010-11-15T10:30+01:00">
```

Nel primo caso l'intervallo di selezione è compreso tra le 13:20 del primo novembre 2010 e le 10:30 del 15 novembre 2010 nel fuso orario del meridiano di Greenwich.

Il secondo esprime il medesimo intervallo nel fuso orario italiano.

## month

In alcuni casi, tutto ciò cui si è interessati è il dato mese/anno. Un classico esempio è quando completiamo una transazione online optando per un pagamento con carta di credito. In tal caso ci viene chiesto di inserire, tra gli altri dati, anche il mese e l'anno di scadenza della carta di credito. Questo controllo ben si presta allo scopo.

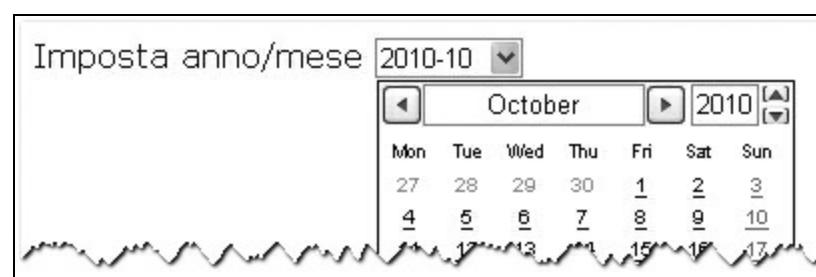
L'elemento produce la stessa rappresentazione grafica del calendario, tuttavia il campo di testo "legato" al calendario verrà popolato, a seguito della nostra selezione, con i soli dati relativi all'anno e al mese prescelto (Figura 4.7).

Il valore predefinito dell'attributo step qui è pari a 1 (mese).

## week

Di sicuro molto meno frequente sembra essere il controllo per la scelta del giorno della

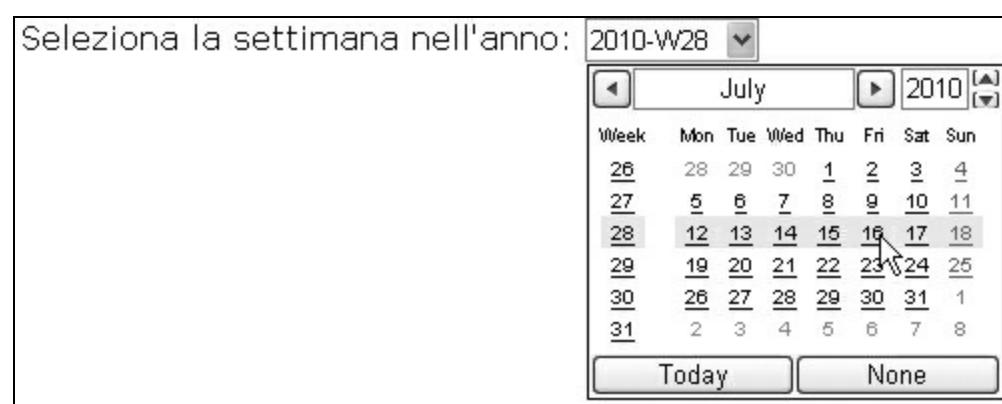
settimana nell'anno. Ancora una volta, il controllo proposto per l'inserimento di questo dato è identico al datepicker illustrato in Figura 4.2.



**FIGURA 4.7** Nel campo di testo sono visualizzati solo anno e mese.

Ciò che cambia è il popolamento del campo di testo collegato: selezionare un qualunque giorno della ventottesima settimana dell'anno, per esempio il 16 luglio 2010, fa sì che l'elemento venga popolato con la stringa “`2010-W28`”.

Il valore predefinito dell'attributo `step` è pari a 1 (settimana).



**FIGURA 4.8** La selezione della settimana è enfatizzata nel controllo da una striscia a sfondo grigio.

## Color picker

Il color picker, alleato imprescindibile di ogni strumento di manipolazione di immagine, è allo stato attuale l'unico controllo ancora non riconosciuto da alcun browser in circolazione. Questo significa che sarebbe sufficiente scrivere:

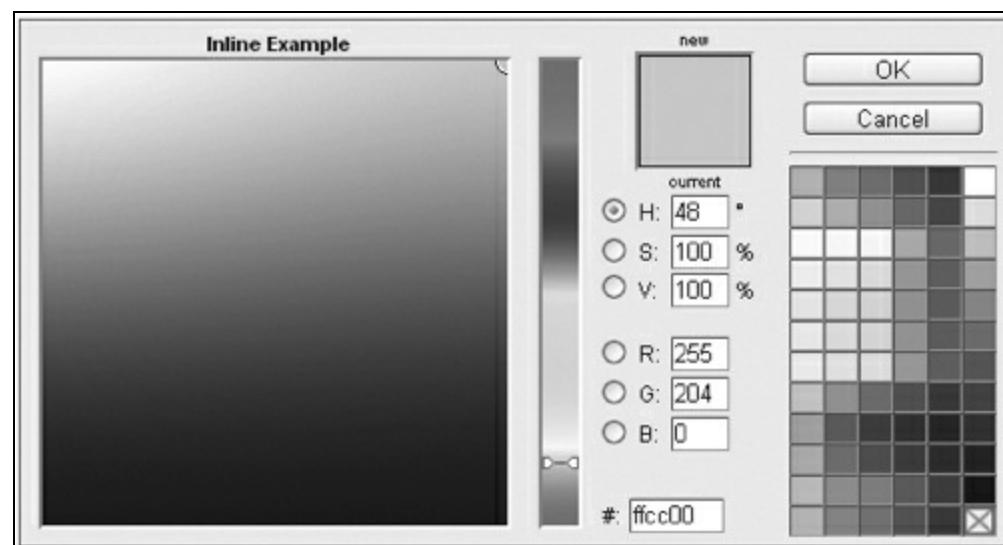
**LISTATO 4.7** Un esempio di color picker

```
<input type="color">
```

per ottenere qualcosa del genere mostrato in Figura 4.9.

Lo scopo dell'elemento è di consentire una facile scelta del colore. La scelta corrisponde a una stringa (una tripletta esadecimale) che è una rappresentazione di quel colore. Poiché questo controllo non è riconosciuto, ciò che in realtà il browser sarà in grado di proporre

all'utente sarà il solito, e in questo caso ancora più tetro, campo di testo, che certo rappresenta un bel passo indietro in termini di usabilità e di gradevolezza dell'interfaccia utente. Anche in questo caso sarà necessario appoggiarsi a librerie JavaScript, in attesa che il controllo sia debitamente interpretato da una vasta gamma di browser.



**FIGURA 4.9** Il color picker.

## Il nuovo elemento `datalist`

Come anticipato all'inizio del capitolo, i nuovi campi input sono solo una parte delle innovazioni prodotte nell'ambito dei form. Il nuovo controllo `datalist` è la soluzione da preferire tutte quelle volte che non è sufficiente scegliere tra un set predefinito di opzioni, perché potrebbe essere utile anche lasciare all'utente "carta bianca".

Prima di HTML5, questa situazione è stata comunemente risolta con due controlli: un combo-box con elenco di opzioni predefinite (il combo-box è altrimenti detto select list o drop down menu; ho sentito persino chiamarlo combo-list o select-box! Sembra che in Italia ci si sia limitati al "vacanziero" menu a tendina: a ogni modo, è possibile affermare con certezza che questo è il controllo con il maggior numero di nomi) e un campo di testo nel caso in cui nessuna delle opzioni proposte soddisfi l'utente.

### **LISTATO 4.8** Quando le opzioni predefinite non bastano!

```
<label for="motivazioni">Cosa ti ha spinto ad acquistare questo manuale?</label>
<select id="motivazioni">
  <option value="riviste_specializzate">Pubblicità su riviste specializzate</option>
  <option value="web_surfing">Ricerche sul Web</option>
  <option value="social_network">Twitter (o altri social network)</option>
  <option value="impulso_distruttivo">Un sano, profondo e incontrollato senso di
  autodistruzione</option>
</select>

<label for="altre_motivazioni">Se altro specificare:</label> <input type="text" id="altre_motivazioni">
```

Per risolvere casi di questo tipo dobbiamo ricorrere alla combinazione di due diversi controlli: un combo-box per le voci predefinite è un campo di testo dove l'utente ha la libertà di specificare un'opzione diversa da quelle pre-impostate.

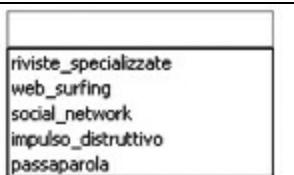
Ecco invece il caso precedente riformulato con il ricorso al nuovo elemento `<data-list>`:

#### LISTATO 4.9 Controllo datalist

```
<label for="altre_motivazioni">Cosa ti ha spinto ad acquistare questo manuale?  
</label>  
<input id="altre_motivazioni" list="motivazioni">  
<datalist id="motivazioni">  
  <option value="riviste_specializzate"></option>  
  <option value="web_surfing"></option>  
  <option value="social_network"></option>  
  <option value="impulso_distruttivo"></option>  
  <option value="passaparola">Passaparola</option>  
</datalist>
```

L'elemento `<datalist>`, di per sé, non viene visualizzato in alcun modo: il suo ruolo si esaurisce nel fornire un elenco preordinato di opzioni tra cui scegliere (Figura 4.10).

Cosa ti ha spinto ad acquistare questo manuale:



**FIGURA 4.10** Il datalist arricchisce il campo di testo offrendo opzioni pre-impostate tra cui scegliere, pur lasciando l'utente libero di proporre una propria scelta.

#### ATTENZIONE

Poiché il controllo `datalist` non si distingue da quello di un campo di testo, si scopre che si tratta di un elemento più ricco solo dopo aver iniziato a inserire dei caratteri (sempre che il primo della sequenza corrisponda a una delle iniziali delle opzioni proposte).

#### Nuovi attributi

Gli undici nuovi attributi sono: `autocomplete`, `autofocus`, `form`, `placeholder`, `list`, `max`, `min`, `multiple`, `pattern`, `required`, `step`. L'attributo `multiple` ha trovato il suo spazio quando si è parlato di email e URL. Si è già trattato di `min`, `max` e `step`, attributi che si applicano tanto ai controlli numerici quanto a quelli relativi a data e ora. Il nuovo elemento `<datalist>` ha dato l'occasione per introdurre l'attributo `list`. Passiamo in rassegna ora, con maggior dettaglio, i rimanenti

attributi.

## autocomplete

La precompilazione è la funzionalità che ripropone la stringa precedentemente inserita in un campo di testo sin dal momento in cui vengono digitati i primi caratteri. Si tratta di uno strumento che velocizza il processo di inserimento dati.

Questa funzionalità è sempre attiva a meno che non si specifichi espressamente il contrario:

- si imposta l'attributo `autocomplete="off"` in relazione al campo `input` su cui si vuole fare a meno dello strumento;
- si imposta l'attributo `autocomplete="off"` sul campo `form` per estendere la disabilitazione a tutti i componenti del `form`;
- si interviene a livello del browser per "sovrascrivere" il comportamento proposto da ogni pagina web con quello desiderato dall'utente.

In ogni caso si tratta di una funzionalità da attivare con cautela, per le implicazioni di riservatezza che emergono nel caso in cui tutti i dati inseriti dall'utente siano sempre ricordati e riproposti. Idealmente si dovrebbe limitarne il campo di applicazione a quei dati considerati poco sensibili, per esempio sarebbe utile astenersi dall'usare la precompilazione per i campi `password`, per i quali si dovrebbe sempre impostare in modo esplicito un `autocomplete="off"`.

## autofocus

L'autofocus è ormai una funzionalità così diffusa che, di norma, si nota solo quando è assente, per la scomodità che comporta dover spostare il puntatore del mouse sino al controllo su cui si desidera operare. Può esistere un solo elemento con autofocus per pagina web. Se questo attributo è applicato a più di un elemento, solo l'ultimo riceverà in concreto l'autofocus. Nel campo di ricerca di Google, per aggiornare lo stato su Twitter o su Facebook e in qualunque interfaccia di login che si rispetti troviamo il cursore lampeggiante proprio lì dove abbiamo bisogno che sia.

### Le due facce dell'usabilità dell'autofocus

A prima vista si potrebbe pensare che non esistano svantaggi nell'uso di questo strumento; del resto sembra che l'autofocus migliori l'interfaccia utente rendendola più amichevole: tutto vero, ma attenzione! Diversi utenti trovano comodo usare la barra spaziatrice per scorrere le pagine web molto lunghe verso il basso. Usare l'autofocus inibisce questo uso della tastiera, che agirà invece sul controllo con l'autofocus inserendo una serie di spazi nel campo di testo cui il focus è stato attribuito.

L'attributo `autofocus` porta nel mondo della marcatura quella che era diventata una prerogativa del linguaggio di scripting. Sebbene si sia abituati a vedere l' autofocus su campi di testo, questo attributo si applica a diversi altri elementi: `button`, `input`, `select`, `textarea`.

Per aggiungere l' autofocus sul campo di ricerca si scriverà:

#### **LISTATO 4.10** Un campo di testo con autofocus

```
<input type="text" id="hobby" autofocus>
```

Per quei browser che, non riconoscendo l'attributo, ne ignoreranno l'impiego, sarà utile continuare a utilizzare il "vecchio" metodo, che consiste nel ricorrere a poche righe di JavaScript nell'intestazione del documento:

#### **LISTATO 4.11** Applicare l' autofocus: "the old way"

```
<script>
// esecuzione da invocare sull'evento onload()
function assegnaFocus() {
  document.getElementById("hobby").focus();
}
</script>
```

### onload() vs DOM ready functions

onload è l'evento JavaScript scatenato quanto la pagina web è stata scaricata in tutti i suoi elementi. Nei casi in cui la pagina sia carica di contenuti che impiegano del tempo per essere visualizzati, per esempio una serie di immagini ad alta definizione, il rischio è di attivare l' autofocus quando l'utente ha già iniziato a interagire con gli elementi del form. In casi come questi, le principali librerie JavaScript offrono una funzione che entra in gioco prima perché viene invocata non appena il DOM HTML (ossia il modello a oggetti del documento) risulta completamente caricato.

Quando si vuole fare affidamento sul più rapido caricamento del DOM si utilizza una delle librerie JavaScript oggi più diffuse. Ecco di seguito come eseguire lo stesso compito (applicare l' autofocus a un campo di ricerca) con tre librerie diverse: Dojo, Prototype e JQuery.

#### **LISTATO 4.12** Autofocus con Dojo Toolkit

```
dojo.ready(function() {
  document.getElementById("hobby").focus();
});
```

#### **LISTATO 4.13** Autofocus con Prototype

```
document.observe("dom:loaded", function() {
  document.getElementById("hobby").focus();
});
```

#### **LISTATO 4.14** Autofocus con jQuery

```
jQuery(document).ready(function() {
  document.getElementById("hobby").focus();
});
```

“Closure” JavaScript library: la grande assente

Closure library, la libreria JavaScript usata da Google in tutte le sue applicazioni web, non presenta niente del genere perché i suoi programmatori ritengono che anche attendere che il DOM sia completamente caricato significhi aspettare troppo! Per questo motivo suggeriscono di usare uno script inline posto subito dopo l’elemento cui si vuole attribuire il focus, come nell’esempio che segue.

#### **LISTATO 4.15** Uno script inline per l’autofocus

```
input type="search" id="hobby" autofocus>
<script>
document.getElementById("hobby").focus();
</script>
```

In alcuni casi questo può non essere possibile per via del framework che stiamo utilizzando, in altri casi si può decidere di evitare questo approccio perché più oneroso in termini di manutenzione: infatti “annega” lo script all’interno del contenuto del documento. Ciò ne rende più difficile l’identificazione per eventuali futuri interventi.

### Autofocus, una strategia alternativa

Per garantire agli utenti che la funzionalità di autofocus sia comunque disponibile, anche in caso di mancato supporto di questo attributo possiamo impiegare la libreria Modernizr. Nel Capitolo 3 abbiamo visto come utilizzare questa libreria per testare il supporto di un elemento, il Listato 4.16 illustra come controllare il supporto di un singolo attributo. Nel caso in cui questo non sia riconosciuto si potrà optare per la soluzione già proposta nel Listato 4.11.

#### **LISTATO 4.16** Test di supporto dell’attributo autofocus con Modernizr

```
<script>
if (!Modernizr.input.autofocus) {
  // in caso di mancato supporto dell'attributo autofocus
  // si richiama una funzione specifica che possa svolgere
```

```
// questo compito. Vedi Listato 4.11
assegnaFocus();
}
</script>
```

Volendo fare a meno di Modernizr si può impiegare la tecnica “fai da te” del Listato 4.17.

#### **LISTATO 4.17** Test di supporto dell’attributo autofocus con tecnica “fai da te”

```
<script>
var input = document.createElement("input");
if (!("autofocus" in input)) {
// attributo non supportato.
// Richiamo funzione JavaScript per attivazione
// 'autofocus per mezzo del linguaggio di scripting
assegnaFocus();
}
</script>
```

Nello script appena proposto si crea dapprima un elemento di tipo `input`, subito dopo si utilizza l’operatore `in`. Esso restituisce il valore booleano `true` se la proprietà, indicata alla sinistra dell’operatore, esiste nell’oggetto indicato alla sua destra, altrimenti restituisce `false`.

## form

Nelle precedenti versioni del linguaggio ogni componente del form si trovava all’interno di una coppia di tag `<form>...</form>`. Esisteva una relazione genitore-figlio dalla quale si desumeva il legame tra ogni componente e il form di appartenenza. Questo comportamento era esplicitamente previsto dalla specifica ufficiale, tanto che ogni tentativo di porre un controllo di un form al di fuori del form stesso avrebbe determinato il fallimento del processo di validazione (Figura 4.11).

Jump To: Validation Output

Error found while checking this document as HTML 4.01 Strict!

Result: 1 Error

Address: Encoding: utf-8 Doctype: HTML 4.01 Strict 

Root Element: HTML

The W3C validators are hosted on server technology donated by HP, and  
supported by community donations.[Donate](#) and help us build better tools for a better web.**Options**

Show Source  Show Outline  List Messages Sequentially  Group Error Messages by Type  
 Validate error pages  Verbose Output  Clean up Markup with HTML Tidy

[Help on the options is available.](#)

Validation Output: 1 Error

✖ Line 12, Column 35: document type does not allow element "INPUT" here; missing one of "P", "H1", "H2", "H3", "H4", "H5", "H6", "PRE", "DIV", "ADDRESS" start-tag

The mentioned element is not allowed to appear in the context in which you've placed it; the other mentioned elements were the only ones that were valid where this one was placed. This might mean that you need a containing element, or possibly that you've forgotten to close a previous element.

**FIGURA 4.11** Il validatore HTML 4.01 segnala la presenza di un controllo del form in una posizione non autorizzata.

Oggi questo non è più necessario. Un elemento può apparire anche all'esterno del form di riferimento pur continuando a esserne parte integrante. Ciò è possibile fintanto che viene esplicitato l'attributo `form` il cui valore sarà pari all'id del form di appartenenza; il Listato 4.18 illustra dunque un documento ben formato secondo il validatore HTML5.

**LISTATO 4.18** Attributo `form`: un esempio

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>hello world</title>
</head>
<body>
<form action="pippo.html" id="questionario">
```

```
<p><input type="text" value="pippo"></p>
</form>
<input type="text" value="pluto" form="questionario">
</body>
</html>
```

## pattern

Questo attributo **apre HTML5 alle “espressioni regolari”**. Paura? Per alcuni è proprio così! Il perché è presto detto; ecco un esempio di un’espressione regolare:

### **LISTATO 4.19** Un’espressione regolare

```
^#?([0-9A-Fa-f]{3}){1,2}$
```

Le espressioni regolari sono lo strumento ideale per il “pattern matching”, ossia per l’identificazione di sequenze di caratteri che seguono uno schema predefinito. Più semplicemente, attraverso le espressioni regolari si può rispondere a domande del tipo: il testo (per esempio quello inserito dall’utente in un campo di testo) rappresenta un codice fiscale? Un codice IBAN? La notazione scientifica di un numero? E via dicendo. L’espressione regolare sopra riportata identifica una tripletta esadecimale secondo uno qualunque dei seguenti formati: #0088Aa, #0088aA, 0088aA, 0088Aa, #08A, 08a.

Possiamo per esempio usare una semplice espressione regolare per verificare che il CAP sia effettivamente composto da cinque numeri:

### **LISTATO 4.20** Espressione regolare applicata al CAP

```
<label for="cap" required>Cap</label>
<input type="text" id="cap" name="cap" pattern="[0-9]{5}">
```

Usando l’attributo `pattern` è possibile snellire il codice di validazione, perché si può rimuovere il codice JavaScript limitandosi a un controllo solo lato server, senza per questo rinunciare a un monitoraggio rigoroso sul browser dell’utente.

Se l’espressione regolare non è soddisfatta (nel caso specifico: se non si inserisce una serie di 5 cifre) il browser può inibire l’inoltro del form e avvisare l’utente. Opera restituisce un messaggio che invita l’utente a rivedere l’inserimento del dato che non soddisfa il pattern previsto dall’espressione regolare (Figura 4.12).

Il tuo recapito

Indirizzo

Via dei Pini, 4

Città

Milano

Cap

2010

2010 non è nel formato richiesto da questa pagina!

**FIGURA 4.12** Il focus è restituito al campo in cui è stato violato il pattern dichiarato.

## Un aiuto agli screen reader

Per consentire alle tecnologie assistive di svolgere pienamente il loro ruolo sarebbe utile inserire una descrizione dell'espressione regolare nel campo `title` che software come Jaws "leggono" per l'utente al fine di fornire un adeguato supporto alla compilazione dei campi.

## placeholder

Il placeholder (letteralmente: "segnaposto") è una parola o una breve frase. In genere si tratta di un testo di esempio oppure una breve descrizione del dato che ci si aspetta l'utente inserisca. Questo testo scompare non appena il campo assume il focus, per poi riapparire se l'utente lascia il campo senza compilarlo.

### **LISTATO 4.21** Un caso d'uso dell'attributo placeholder

```
<form method="post" action="http://html5.gigliotti.it/process_data.php">  
<fieldset>  
  <legend>I tuoi dettagli</legend>  
  <p>  
    <label for="first_name">Il tuo nome</label>  
    <input type="text" id="first_name" name="first_name" size="30" placeholder="Inserisci nome e cognome">  
  </p>  
  <p>  
    <label for="tel">Telefono</label>  
    <input type="tel" id="tel" name="tel" size="30" placeholder="Inserisci il numero di telefono">  
  </p>
```

```

<p>
  <label for="email">Email</label>
  <input type="email" id="email" name="email" size="30" placeholder="tua@mail.it">
</p>
<p>
  <label for="password">Password</label>
  <input type="password" id="password" name="password" size="30" placeholder="Lunghezza minima 6
caratteri">
</p>
<p>
  <input type="submit" name="invia" value="Invia!">
</p>
</fieldset>
</form>

```

Il listato produce su Chrome il risultato visibile in Figura 4.13. Il placeholder può essere utilizzato tanto come semplice etichetta, quanto come ulteriore supporto all'utente. Per esempio, nel campo password si invita l'utente a scegliere una password che non sia di lunghezza inferiore ai 6 caratteri. In questo caso si usa il placeholder per aiutare chi compila il form con suggerimenti e indicazioni proposte esattamente lì dove servono.

The screenshot shows a web form with the following fields and their placeholder text:

- I tuoi dettagli** (Section title)
- Il tuo nome** (Text label):
- Telefono** (Text label):
- Email** (Text label):
- Scegli la tua password** (Text label):
- Invia!** (Submit button)

**FIGURA 4.13** Il placeholder secondo Chrome.

## La giusta lunghezza

È bene fare uno sforzo di sintesi quando si vuole aggiungere un placeholder. Se si deve scrivere un testo più lungo è meglio ricorrere all'attributo `title`.

## required

Contrassegnare un campo con questo attributo equivale a identificarne il contenuto come obbligatorio ai fini del corretto e completo inoltro dei dati. È inutile dire che ricorrere a questo attributo permette di rafforzare la semantica del documento oltre a rendere esplicita, in termini di linguaggio di marcatura, la distinzione tra un campo obbligatorio e uno la cui compilazione è opzionale. Non è escluso che gli screen reader in futuro possano sfruttare questa informazione per una più efficace assistenza nella compilazione del form. Oggi è prassi contraddistinguere i campi obbligatori di un form con un asterisco. Il controllo del rispetto della compilazione di questi campi avviene poi, di norma, in due tempi: lato client usando JavaScript in modo da dare una risposta immediata all'utente in caso di omissioni nella compilazione e, ovviamente, lato server.

## Altri marcatori

Lo studio dei nuovi elementi e relativi attributi che ha avuto inizio nel precedente capitolo termina qui con i due marcatori `<keygen>` e `<output>`.

### <keygen>

Con HTML5 diventa parte della specifica un elemento già da tempo in uso presso i browser di casa Netscape. `<keygen>` è un controllo espressamente concepito per la generazione di coppie di chiavi pubbliche e private impiegate nei sistemi di crittografia asimmetrica. La specifica prevede espressamente che all'atto dell'inoltro dei dati del form la chiave privata sia conservata localmente mentre quella pubblica sia criptata e inoltrata al server. Due attributi sono specifici di questo marcitore:

- **challenge**: è una stringa che può essere specificata in aggiunta alla chiave con scopo di controllo dell'autenticità della medesima;
- **keytype**: identifica la tipologia di algoritmo di crittografia utilizzata nel processo di generazione della chiave. Il solo valore ammesso è, a oggi, `RSA` (acronimo ricavato dai nomi dei tre inventori dell'algoritmo: Rivest, Shamir e Adleman).

### <output>

Questo marcitore esprime il risultato di un calcolo. Il solo attributo specifico per questo elemento prende il nome di `for` e accetta uno o più `id` - separati da virgola - che identificano altri elementi i cui relativi valori hanno contribuito alla determinazione del risultato marcato con `<output>`.

## Alcuni consigli

La raccolta dati degli utenti, compiuta attraverso i form web e la relativa copia in database sempre più ricchi di informazioni, deve essere compiuta con quel giusto equilibrio tra il legittimo desiderio di sondare sin nei minimi particolari vizi e virtù dei propri utenti per ottenerne una profilazione accurata e l'imprescindibile raccolta solo di quei dati strettamente necessari a erogare il servizio.

## Tutela dei dati personali

La prassi ormai diffusa, dentro e fuori il Web, di eccedere nella richiesta di informazioni è un tema dibattuto da molto tempo. L'etica hacker di Pekka Himanen (Feltrinelli editore) affronta il dissenso espresso sul tema da alcuni hacker statunitensi.

Qualunque sia la scelta che si decide di compiere, è sempre necessario porre l'utente nelle condizioni di completare il compito con la massima semplicità possibile.

Come nella tradizione degli "avvertimenti" in stile statunitense (della serie: "attenzione, il caffè è molto caldo, potresti ustionarti", "attenzione, gli oggetti che vedi nello specchietto retrovisore sono più vicini di quanto non sembrino"), inizio questo elenco con il seguente disclaimer: "Attenzione: i suggerimenti di seguito elencati rappresentano un condensato di buon senso, esperienza e delle migliori pratiche adottate sul tema". Prima di passare ai vip di HTML5, quei marcatori video e audio che tanto hanno fatto parlare, ci si prenda qualche minuto di tempo per passare in rassegna i seguenti punti.

- Anche se siete giovani Harry Potter del web design e i vostri form sono accattivanti come nessun altro, dovete rassegnarvi davanti a una disarmante verità: tutti, non ci sono eccezioni, odiano compilare un form. Nessuno vuole perdere del tempo a indicare nome, cognome, età, indirizzo ecc. Non c'è niente di affascinante in tutto ciò! Per questo motivo, cercate di ridurre il numero degli elementi del form ai minimi termini. Alla larga da chi ti invita a inserire il tuo curriculum in 23 comodi step! Per completare questo lavoro di sintesi, mettetevi nei panni dell'utente e domandatevi se il dato che gli si chiede è utile. Se non lo è, mettetelo da parte; può darsi che riusciate a convincere l'utente in un secondo momento, sempre che siate in grado di dimostrarigli il valore di questo sforzo aggiuntivo che gli state chiedendo. In questo senso LinkedIn fornisce una percentuale di completamento del proprio profilo. Se l'utente fornisce più dati, compilando sezioni aggiuntive del form, il profilo diventa più esaustivo e più interessante per potenziali contatti.
- Per aumentare il tasso di conversione tra chi decide di compilare il form con i suoi dati e chi effettivamente arriva in fondo, inoltrando concretamente il form, dovete limitarvi a chiedere informazioni di facile reperibilità. Se, per compilare il form, devo: a) alzarmi dalla sedia b) scartabellare tra i miei documenti c) ritornare alla scrivania per terminare l'operazione, per poi dover rimettere a posto le scartoffie, ci sono elevate probabilità che io desista!

- Se dovete invitare l'utente a compiere una scelta tra più opzioni note, usate un controllo che preveda la scelta tra voci predefinite onde evitare un doloroso, in termini di tempi e costi, lavoro di normalizzazione dei dati lato server. Per esempio, sono bastati pochi minuti sul Web per trovare almeno 7 modi diversi di indicare l'Università degli Studi di Milano (la Statale, Univ. Statale Mi, Univ. Statale Milano, Univ. Stat. Milano, Università di Milano Statale, Statale di Milano, Unimi).
- **Mai adottare politiche di validazione diverse tra client e server.** Per esempio, se si richiede l'inserimento del codice fiscale, sarà fondamentale applicare lo stesso algoritmo di validazione in entrambi gli ambienti, come usare lo stesso pattern di un'espressione regolare, per evitare inconsistenze nel comportamento dell'applicativo che riconosce valido un dato in un ambiente (per esempio lato client) per poi ritenerlo errato in un altro ambiente (per esempio lato server).
- Se l'utente è costretto a ricompilare uno o più campi, è necessario mostrare come precompilati tutti gli altri in cui l'informazione è stata inserita in modo corretto, limitando il reinserimento dei dati solo in casi ben precisi (per esempio l'inserimento della password). Sebbene questo sia lapalissiano, perché l'utente difficilmente compilerà di nuovo dodici campi solo per un errore nel tredicesimo, accade spesso che la precompilazione avvenga per i campi di testo e, solo di rado, per esempio, per caselle di controllo!
- **Abiate cura di usare l'attributo `autofocus`** sul primo elemento del form ricordandovi di adottare una strategia alternativa appropriata, come quelle discusse in questo capitolo, per i browser che non riconoscono l'attributo. A questo punto l'utente può dimenticare il mouse e spostarsi da un controllo al successivo usando la sola tastiera. Applicare l'autofocus al primo elemento di un form è già un invito al completamento dello stesso. Un campo di ricerca (vedi Google, Bing, Yahoo), un form di login, un form di inserimento dati (quale quello tipico di un processo di acquisto online) sono tutti casi d'uso che ben si prestano all'impiego di questo attributo.
- **Da evitare ove possibile: il bottone reset**, utilizzato solo di rado, se premuto accidentalmente costringe l'utente a ripartire da zero.
- Il combo-box che permette selezioni multiple richiede sempre la didascalia (su Win occorre tenere premuto il tasto Ctrl e su Mac il tasto Command); inoltre non è il massimo in termini di usabilità perché quando si selezionano due o più opzioni non contigue c'è il rischio di deselectare accidentalmente le altre scelte senza rendersene conto.
- **Insiemi di controlli logicamente collegati** (per esempio i campi relativi all'indirizzo, quelli relativi alla carta di credito o ancora un gruppo di caselle di controllo che rappresentano possibili opzioni di risposta a una stessa domanda) dovrebbero essere racchiusi da un marcitore `<fieldset>`. Esso migliora la semantica del documento rendendo esplicita la relazione tra i campi.
- **Usate l'elemento `<label>`** per contrassegnare le etichette dei campi. Tale marcatore

è utile per diversi motivi: **rende più efficace il lavoro svolto dallo screen reader** perché garantisce la corretta associazione tra etichetta e controllo, migliora la semantica del documento perché contrassegna in modo esplicito la relazione tra etichetta e controllo. Nelle caselle di controllo (checkbox) e nei pulsanti d'opzione (radio button), laddove bisogna essere più precisi con il puntatore per fare clic esattamente sull'area ridotta che il controllo mette a disposizione, l'etichetta del campo diventa essa stessa parte del controllo. Facendo clic sull'etichetta della casella di controllo o del pulsante d'opzione, l'elemento corrispondente risulta selezionato.

- **I campi obbligatori devono essere immediatamente identificabili rispetto a quelli opzionali.** Un asterisco di fianco all'etichetta, o la stessa etichetta in grassetto, sono i due modi più diffusi per applicare questa distinzione.
- Alle volte si fa confusione nella scelta dei controlli più adatti. Le difficoltà maggiori si incontrano quando si confonde il combo-box con la casella di controllo o il pulsante di opzione. Si scelga il combo-box quando le opzioni sono parecchie: sfortunatamente non esistono numeri precisi, poiché molto dipende dal layout del documento. Alcuni dicono di ricorrere al combo-box solo oltre le 10 opzioni, altri suggeriscono 5 opzioni come numero massimo. La verità è che non esiste una soglia precisa: ci si farà guidare dal buon senso e dalla prassi (per esempio non mi sembra di aver mai visto l'elenco delle regioni o delle province italiane proposte come elenco di pulsanti di opzione). Si userà il pulsante di opzione quando la scelta che si chiede di eseguire è mutuamente esclusiva (si pensi per esempio alla richiesta di selezionare una fascia di età: la triste realtà è che indipendentemente da quanto ci si senta giovani, ognuno di noi appartiene a una data fascia di età in ogni dato momento e questo lo esclude da tutte le altre). Un gruppo di caselle di controllo è utile solo se ha senso che si possano selezionare più scelte. Per una scelta tra due opzioni mutuamente esclusive si può adottare tanto una coppia di pulsanti d'opzione, quanto la singola casella di controllo. Buon lavoro!

## Riferimenti alle risorse citate e altri link utili

- Controlli relativi a data e ora – la specifica W3C: <http://www.w3.org/TR/html5/states-of-the-type-attribute.html#date-and-time-state>
- Attributo `autocomplete` – la specifica W3C: <http://www.w3.org/TR/html5/common-input-element-attributes.html#the-autocomplete-attribute>
- Attributo `autofocus` – la specifica W3C: <http://www.w3.org/TR/html5/association-of-controls-and-forms.html#attr-fe-autofocus>
- **L'importanza della funzionalità di `autofocus`** è tale da aver spinto persino alla creazione di un plug-in (un componente aggiuntivo) dedicato per la libreria jQuery: <https://github.com/miketaylr/autofocus>. Questo componente interviene attivando

la funzionalità di autofocus laddove l'attributo medesimo non sia supportato.

- Attributo `form` – la specifica W3C: <http://www.w3.org/TR/html5/association-of-controls-and-forms.html#association-of-controls-and-forms>
- Attributo `pattern` – la specifica W3C: <http://www.whatwg.org/specs/web-apps/current-work/multipage/common-input-element-attributes.html#the-pattern-attribute>
- Definizione di “espressione regolare” in Wikipedia: [http://it.wikipedia.org/wiki/Espressione\\_regolare](http://it.wikipedia.org/wiki/Espressione_regolare)
- Attributo `placeholder` – la specifica W3C: <http://www.whatwg.org/specs/web-apps/current-work/multipage/common-input-element-attributes.html#the-placeholder-attribute>
- Attributo `required` – la specifica W3C: <http://www.whatwg.org/specs/web-apps/current-work/multipage/common-input-element-attributes.html#the-required-attribute>
- La definizione dell’algoritmo RSA pubblicata da Wikipedia: <http://it.wikipedia.org/wiki/RSA>
- Se la crittografia è un argomento che vi appassiona, c’è un libro che non potete lasciarvi scappare: Codici & segreti. La storia affascinante dei messaggi cifrati dall’antico Egitto a Internet, di Simon Singh: [http://www.lafeltrinelli.it/products/9788817125390/Codici\\_segreti/Singh\\_Simon.html](http://www.lafeltrinelli.it/products/9788817125390/Codici_segreti/Singh_Simon.html)

## Conclusioni

La specifica un tempo nota come Web Forms 2.0, e ora diventata parte integrante della specifica HTML5, semplifica il lavoro degli autori di pagine web e rende più gradevole l’esperienza d’uso dei visitatori: due buoni motivi per utilizzare sin d’ora le nuove funzionalità. Tuttavia, non si possono nascondere le lacune di supporto che affliggono questa parte della specifica, se è vero che solo Opera, browser validissimo ma pur sempre con quote di mercato largamente minoritarie, può vantare un supporto quasi completo in quest’area. Eppure sarebbe un errore ancora più grande bollare queste funzionalità come immature e optare per ignorarle in attesa di un supporto più esteso. Uno tra i siti più popolari al mondo come Amazon.com, nato come libreria online e trasformatosi ben presto nel negozio di vendita al dettaglio sul Web più fornito di tutti gli Stati Uniti, ha colto le potenzialità di strumenti che consentono di offrire un’esperienza più ricca ai propri utenti. In Figura 4.14 si evidenzia come nel form di autenticazione, in cui Amazon.com invita a inserire le proprie credenziali, il campo di testo finora utilizzato per ospitare l’indirizzo e-mail dell’utente sia stato sostituito con un campo di tipo email. In questo modo il sito giova del controllo di validità dell’indirizzo di posta elettronica incluso nel browser. Un ulteriore vantaggio di questa scelta è quello di poter fornire un messaggio di errore nella lingua dell’utente (infatti è il browser, non l’applicativo web di Amazon.com, che si preoccupa di visualizzarlo).



**FIGURA 4.14** Amazon utilizza il campo <input type="email"> nel suo form di autenticazione.

# Video & audio: la lingua franca del Web diventa multimediale

Video e audio HTML5 sono i due elementi che, sin dal primo momento, hanno riscosso maggiore interesse sulla Rete. Non mi riferisco al legittimo entusiasmo di un'oscura nicchia di addetti ai lavori, ma al gran numero di importanti aziende che ne hanno subito dichiarato l'adozione in alternativa alle soluzioni proprietarie che finora hanno dominato la scena del multimedia sul Web; tra queste, Adobe Flash più di ogni altra. Un esempio su tutti è quello di Apple; Steve Jobs, in una lettera aperta intitolata "Considerazioni su Flash", spiega quali sono i motivi che hanno portato Apple a optare per il video HTML5. Anche altre aziende hanno creduto in questa tecnologia: YouTube, che a marzo del 2010 ha raggiunto il totale di 24 ore di video caricati sui suoi server ogni 60 secondi, ha investito molto su di essa, se si pensa ai diversi formati in cui è necessario rendere disponibile un video HTML5, in aggiunta alla scelta di dimensioni offerte per ciascun filmato. Amazon ha lanciato da poco il suo "Kindle per il Web", una piattaforma per leggere e condividere gratuitamente anteprime di libri a pagamento. HTML5 e CSS3 sono dichiaratamente le tecnologie scelte per il lancio di questa comunità di utenti.

A New York si è tenuta una serie di eventi sul tema dell'evoluzione multimediale nell'ambito della neo-nata Open Video Conference svoltasi a ottobre 2010: ebbene, molti dei seminari hanno avuto a oggetto temi legati al nuovo elemento `<video>` HTML5.

È corretto annotare che da più parti sono state sollevate perplessità sulla specifica nel suo complesso e, più in particolare, sul video nativo HTML5; in tale ambito il linguaggio si presenta come un'alternativa ancora poco credibile rispetto ad altre tecnologie, perché inadatta a proteggere adeguatamente i contenuti e ancora impreparata all'integrazione con le inserzioni pubblicitarie.

Tuttavia, nell'arco di pochi mesi sono state sviluppate diverse librerie volte a uniformare i controlli di interfaccia utente (il cosiddetto look & feel) che, come messo in evidenza anche dalle immagini proposte in questo capitolo, presentano una stilizzazione diversa in base al browser e al sistema operativo. Inoltre, tali librerie JavaScript mirano a rimuovere problemi di compatibilità con vecchi browser che non riconoscono i tag `<audio>` e `<video>`: è questo il caso di [mediaelementjs.com](http://mediaelementjs.com) o [videojs.com](http://videojs.com). Altre librerie invece si pongono l'obiettivo di migliorare l'accessibilità di risorse audio e video; [PopCorn.js](http://PopCorn.js) (sviluppata dall'Open Video Lab di Mozilla) permette per esempio di aggiungere dei sottotitoli ai propri filmati, anche in diverse lingue, dando l'opportunità all'utente di cambiarli al volo. Nonostante il generale interesse suscitato da un contesto multimediale strettamente

interconnesso con le altre tecnologie web come CSS e JavaScript, esistono ancora alcune criticità ancora da superare: la tabella 5.4 sintetizza i principali elementi pro e contro da valutare attentamente prima di decidere per l'adozione di questa tecnologia. La cosa più importante è di non lasciarsi ingabbiare nello schema `<video>` HTML5 contro Flash. Le tecnologie vivono un diverso stato di maturità, presentano caratteristiche diverse e, pur presentando aree di sovrapposizione, possono essere impiegate entrambe con profitto a seconda dei propri obiettivi.

Infine, la guerra, appena iniziata, dei codec (il formato di compressione/decompressione con cui i video HTML5 possono essere visualizzati) potrebbe ostacolare la diffusione dello strumento. Oltre a ciò, gli elementi audio e video dovranno evolvere in fretta, offrendo funzionalità che vadano ben oltre quelle basilari, se vorranno proporsi come valide alternative a quelle tecnologie proprietarie che oggi si sentono minacciate.

**TABELLA 5.1** Supporto degli elementi `<video>` e `<audio>`

<b>ELEMENTO SUPPORTATO</b>	<b>BROWSER</b>
<code>&lt;audio&gt;</code>	Chrome 3+, Firefox 3.5+, IE9, Opera 10, Safari 3.2+
<code>&lt;video&gt;</code>	Chrome 3+, Firefox 3.5+, IE9, Opera 10, Safari 3.2+

## L'elemento `<video>`

Nella sua forma più semplice, l'elemento `<video>` ricorda da vicino il tag `<img>`, utilizzato per inserire un'immagine nel documento.

**LISTATO 5.1** Sintassi base di un elemento `<video>`

```
<video src="video.webm">
```

Nella realtà dei fatti, tuttavia, è altamente improbabile che si usi il marcatore `<video>` in questo modo, se non altro per la scomodità di accesso alle funzionalità base (play, pausa, loop ecc.) che, in questo caso, possono essere accessibili solo attraverso un menu contestuale del browser (il comportamento di Chrome è illustrato in Figura 5.1). Più di ogni altra cosa, questa sintassi è scoraggiata perché costringe a offrire formati diversi dello stesso video se ci si vuole garantire la ragionevole certezza della fruizione del filmato presso i propri utenti.



**FIGURA 5.1** In assenza di comandi per intervenire sul video, l'interazione è possibile mediante un menu contestuale.

I controlli attraverso i quali, per esempio, avviare o interrompere l'esecuzione del filmato sono un primo passo verso un'interfaccia utente più usabile.

**LISTATO 5.2** Sintassi base di un elemento <video>

```
<video src="video.webm" controls>
```

L'attributo booleano `controls` assolve questo compito esponendo comandi e informazioni utili a tenere sotto controllo l'esecuzione del filmato (Figura 5.2).



**FIGURA 5.2** I controlli standard messi a disposizione da Chrome su Windows XP grazie all'ausilio dell'attributo `controls`. I controlli sono sempre visibili.

Internet Explorer 9, prima versione del browser di casa Microsoft a implementare il marcitore `<video>`, ignora l'assenza dell'attributo `controls` e mostra comunque i controlli al passaggio del puntatore del mouse sull'area occupata dal video. Per questo motivo il suo menu contestuale presenta solo un sottoinsieme delle voci che propone Chrome (Figura



**FIGURA 5.3** Internet Explorer 9 mostra i comandi al passaggio del mouse sull'area del video indipendentemente dalla presenza dell'attributo controls.

La stilizzazione dei controlli varia a seconda del browser e del sistema operativo, la disposizione dei comandi rimane pressoché inalterata. Da sinistra verso destra troviamo:

- il pulsante Play, per lanciare l'esecuzione del filmato; si tramuta in un pulsante Pausa a esecuzione in corso;
- la barra con indicatore scorrevole che misura il tempo ed è utile per posizionarsi su un punto arbitrario del filmato;
- un contatore minuti:secondi che arricchisce l'informazione fornita dall'indicatore; Firefox sarà in grado di fornire informazioni relative alla durata complessiva del filmato solo se il server supporta uno specifico tipo di "request" detta byte range;
- uno slider, disponibile al passaggio del mouse, per regolare l'intensità del volume (Figura 5.4).



**FIGURA 5.4** Lo slider per la regolazione del volume compare al passaggio del mouse.

Siamo ancora lontani dall'interfaccia evoluta con cui siamo abituati a interagire, dove oltre ai controlli base sopra indicati è anche possibile cambiare risoluzione, attivare/disattivare i sottotitoli o accedere facilmente alla modalità di visione a pieno schermo o variare la velocità di visione (avanti/indietro veloce).

Opera propone i propri controlli solo al passaggio del mouse o su focus che l'elemento riceve da tastiera (sì, su Opera è possibile interagire con il video anche da tastiera spostandosi con il tasto Tab da un controllo al successivo!).



**FIGURA 5.5** In Opera si nota una stilizzazione diversa dei controlli.

Safari 5 su Mac propone un controllo ulteriore, che consente di passare a una visualizzazione a tutto schermo (con il pulsante all'estrema destra in Figura 5.6).



**FIGURA 5.6** L'icona composta da due frecce posta all'estrema destra della barra dei controlli serve per offrire un'immagine del video a tutto schermo.

L'altro attributo fin qui utilizzato, oltre a `controls`, è `src`, che serve a specificare l'indirizzo del file, comportandosi esattamente come accade per l'elemento `<img>`. Più avanti vedremo come sia possibile fare a meno di questo attributo facendo ricorso, in sua vece, all'elemento `<source>`.

Ora passiamo in rapida successione gli altri attributi supportati dal tag `<video>`.

## Attributo poster

L'immagine che il video mostra prima della sua esecuzione è data dal primo fotogramma, che nell'esempio mostrato in queste pagine è un coloratissimo tucano di legno. Qualora si pensi che tale immagine non sia rappresentativa del filmato, si può sostituirla mediante

l'uso dell'attributo `poster`. Così facendo si può specificare l'indirizzo del file che verrà proposto come "copertina" di questo video.

### LISTATO 5.3 Come impostare un'immagine iniziale diversa dal primo frame del filmato

```
<video src="video.webm" poster="libreria.jpg">
```

## Attributo `preload`

In precedenti versioni della specifica, questo attributo aveva il nome di `autobuffer`: un attributo booleano che, se esplicitato, aveva la funzione di iniziare a caricare il contenuto del video anche senza espressa indicazione dell'utente. `autobuffer` è stato soppiantato da `preload`.

Non si tratta di un semplice cambio di nome; ora l'attributo può assumere 3 stati.

1. `none`: nessun caricamento della risorsa deve avere luogo, almeno fino al momento in cui l'utente non decide di lanciare l'esecuzione del video. Se ne suggerisce l'utilizzo nei casi in cui l'autore del documento ritenga improbabile si voglia prendere visione del filmato, o anche solo per ridurre il consumo di banda (aspetto, questo, sempre da valutare con attenzione per l'utenza mobile).
2. `metadata`: anche in questo caso il browser non procede al caricamento della risorsa (a meno che non sia impostata un'immagine da proporre al posto del primo frame del video, nel qual caso verrà recuperato quest'ultimo) ma si preoccupa di rendere disponibili alcune informazioni relative al filmato: durata, dimensioni, prima immagine del video.
3. `auto`: in questo caso il browser cercherà di eseguire il caricamento della risorsa. In altri termini, si sta assumendo come altamente probabile che l'utente voglia prendere visione del filmato. In un tale contesto, il pre-caricamento del video ha lo scopo di ridurre o annullare ritardi dovuti all'acquisizione dell'intero file. Se si omette il valore per l'attributo `preload`, `auto` è da considerarsi il suo valore predefinito. Pertanto, scrivere `<video src="video.mp4" preload="auto">` equivale a digitare `<video src="video.mp4" preload>`. Se avete a cuore la leggibilità del codice scegliete la prima sintassi, se siete amanti della sintesi e della riduzione spasmodica dei byte a ogni costo optate per la seconda. L'importante, come sempre, è che siate coerenti: una volta adottato un approccio, continuate a seguirlo.

## Attributi `autoplay` e `loop`

Come lo stesso nome lascia intuire, l'attributo booleano `autoplay` scatena l'esecuzione del video automaticamente.

```
<video src="video.webm" autoplay>
```

Qui è necessario prestare attenzione ad almeno un paio di considerazioni.

- Alcuni utenti potrebbero considerare invadente la scelta di far partire l'esecuzione del filmato senza un'esplicita azione. A chi obietta che se l'utente ha raggiunto la pagina web su cui si trova il video è legittimo aspettarsi voglia prenderne visione, si può facilmente rispondere che i modi per raggiungerla sono, per definizione, imprevedibili.
- Parafrasando lo slogan “Consuma alcol responsabilmente”, si potrebbe scrivere: “Consuma banda responsabilmente”: i possessori di smartphone te ne saranno grati, soprattutto se hanno scelto un piano tariffario a consumo!

## Autoplay e preload

Come sottolinea la stessa specifica HTML5, gli attributi `autoplay` e `preload` presentano aree di sovrapposizione, perché è corretto osservare che il filmato deve essere caricato prima di poter essere eseguito. Impostare contemporaneamente tanto l'esecuzione automatica quanto il pre-caricamento, sebbene non sia da considerarsi un errore, è certamente ridondante.

Anche l'attributo `loop` è booleano. Sarà dunque sufficiente indicarlo senza attribuirvi alcun valore per ottenere che l'esecuzione del video riprenda non appena ha completato la sua esecuzione.

## Attributi width e height

Anche in questo caso, l'uso degli attributi è sin troppo evidente: il loro scopo è di definire larghezza e altezza del video. Qualora queste informazioni non siano disponibili, sarà onere del browser calcolarle, tenendo conto che, qualora si sia specificata un'immagine da proporre al posto del primo frame del filmato, saranno le dimensioni di quest'ultima a determinare quelle del video.

```
<video src="video.webm" width="640" height="480">
```

## Elemento <source>

Finora abbiamo sempre utilizzato l'attributo `src` per proporre l'indirizzo del filmato. Analizziamo ora i vantaggi di una sintassi alternativa, che prevede l'impiego dell'elemento

<source>.

## **LISTATO 5.6** Utilizzo dell'elemento <source> per l'identificazione dell'indirizzo del filmato

```
<video controls>
  <source src="video.estensione" type="mime type e codec utilizzati">
</video>
```

Qui l'attributo `src` è stato sostituito dall'elemento `<source>`. Una sintassi più verbosa come questa consente di fornire più elementi `<source>`, ciascuno dei quali verrà utilizzato per proporre un diverso formato video. Ciò aumenta le probabilità che il browser disponga del lettore adatto ad almeno uno dei formati messi a disposizione o direttamente o attraverso un componente esterno.

## **LISTATO 5.7** Elemento <video> con sorgenti in formati diversi

```
<video controls>
  <source src="video.webm" type='video/webm; codecs="vp8, vorbis"'>
  <source src="video.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
  <source src="video.ogv" type='video/ogg; codecs="theora, vorbis"'>
  <p>Se non visualizzi il filmato puoi scaricarlo in formato <a href="video.webm">WebM</a>,
    <a href="video.mp4">mp4</a> o <a href="video.ogv">Ogg Theora</a></p>
</video>
```

Sono tre i formati proposti in questo esempio: WebM, Mp4 e Ogg, che sono poi quelli più diffusi. Il seguente box "La guerra dei codec" offre una panoramica delle scelte strategiche adottate dai principali browser, che in estrema sintesi sono riportate in Tabella 5.2.

## **TABELLA 5.2** Codec e relativi formati supportati dai principali browser

<b>CODEC VIDEO</b>	<b>CODEC AUDIO</b>	<b>FORMATO</b>	<b>BROWSER</b>
VP8	Vorbis	WebM	Chrome, Firefox4, Opera10, IE9*
Theora	Vorbis	Ogg	Chrome, Firefox3.5, Opera10
H.264	AAC	MP4	Chrome, IE9

\* Internet Explorer 9 dichiara il supporto per il codec video VP8 solo se installato separatamente sul sistema operativo.

## La guerra dei codec

Con una mail dell'11 dicembre 2007, Ian Hickson, editore dell'HTML5, dichiarò di avere temporaneamente rimosso

dalla specifica i requisiti per la definizione del formato di compressione/decompressione (codec, appunto) dei video. Ciò accadeva per via delle difficoltà nel trovare un accordo tra le parti su quale formato usare. Sono trascorsi 3 anni, ma questa intesa non è stata raggiunta. Di fatto, i browser più importanti hanno compiuto e stanno portando avanti scelte diverse.

Il formato video può essere definito, seppure con qualche approssimazione, come una sorta di contenitore al cui interno è possibile trovare una traccia video, una traccia audio e un file contenente metadati, ossia informazioni relative al filmato stesso.

Internet Explorer 9 e Safari hanno optato per il supporto al codec H.264, noto come standard del Blu-ray proposto attraverso il formato Mp4. Scelta diversa è stata quella di Firefox e Opera, che invece prediligono i formati Ogg/Theora e WebM (formato aperto e ottimizzato per il Web che funge da contenitore per il codec audio Vorbis e il codec video VP8). Chrome li supporta tutti e tre. Internet Explorer 9 supporta il codec VP8 solo se questo risulta installato separatamente sul sistema operativo.

La Mpeg Association che detiene le licenze H.264 si è subito mossa per contrastare la rapida diffusione del formato aperto WebM, decidendo di offrire gratuitamente sul Web il proprio codec; tuttavia non è chiaro quali siano i termini entro i quali questa licenza può essere impiegata in modo gratuito. La decisione di Microsoft di non proporre come pre-installato il codec VP8 rende di fatto necessario l'impiego di almeno due codec (WebM e Mp4) per coprire ragionevolmente tutti i browser di nuova generazione. Ce n'è abbastanza per poter dire che una guerra di posizionamento è in atto. Il timore è che tutte queste frammentazioni finiscano per ostacolare l'adozione e la diffusione del video in HTML5.

L'elemento `source` presenta due attributi: `src` e `type`.

Come ci si aspetterebbe, `src` ha lo scopo di fornire l'indirizzo del filmato che, nell'esempio sopra riportato, si troverà nella stessa cartella della pagina web.

L'attributo `type` invece svolge la duplice funzione di definire il formato del filmato e di indicare la compressione audio e video adottata nell'ambito di tale formato. Se ne desume che questa dichiarazione riveste particolare importanza; eccone dunque alcuni esempi, con la relativa spiegazione.

#### **LISTATO 5.8** Codec Video Theora e codec audio Vorbis proposti in un contenitore (formato) Ogg

```
<source src="video.ogv" type='video/ogg; codecs="theora, vorbis"'>
```

#### **LISTATO 5.9** Nessun codec video. Codec audio FLAC audio in formato Ogg

```
<source src="audio.oga" type="audio/ogg; codecs=flac">
```

#### **NOTA**

L'estensione `ogg` usata in passato è ora deprecata in favore di `ogv`, per file video, e `oga` per file audio. Una disamina più approfondita sul tema è disponibile presso il sito della fondazione Xiph.org, nell'ambito della quale tale formato è stato sviluppato.

#### **LISTATO 5.10** Codec Video H.264 in alta qualità livello 3 e codec audio AAC in formato MP4

```
<source src='video.mp4' type='video/mp4; codecs="avc1.64001E, mp4a.40.2"'>
```

Il formato mp4 è, tra tutti, quello che presenta il maggior numero di combinazioni tra codec audio e video utilizzabili al suo interno. La specifica HTML5, a solo titolo di esempio, ne propone 6 diversi!

Omettere l'attributo `type` non compromette la fruizione del filmato, ma complica il lavoro del browser e richiede un dispendio maggiore di banda. Infatti, per sapere quale, tra quelli proposti, è il formato fruibile dal browser, quest'ultimo dovrà passarli in rassegna uno a uno leggendone una piccola parte per interpretarne la codifica.

## Ricordarsi di configurare appropriatamente il server

Tutta la nostra precisione nel definire con cura codec e formato servirà a poco se non abbiamo l'accortezza di configurare sul server il MIME Type corrispondente per questo tipo di file. Su server Apache è sufficiente modificare il file `httpd.conf` aggiungendo i nuovi tipi con la direttiva `AddType`. Per esempio: `AddType video/webm .webm`.

Alla fine, se esiste un formato idoneo a essere riprodotto l'utente potrà vedere il filmato, pagando però un extra costo di banda, necessario per questa "caccia" al formato. Faccio una previsione: gli utenti mobile con piani tariffari a traffico non faranno i salti di gioia, se a questo sommiamo i ritardi generalizzati che fiaccano lo stato delle ADSL in Italia, uno dei paesi con il più basso tasso di penetrazione della banda larga in tutta Europa (La Stampa del 27 settembre 2010: "Altro che banda larga: gli utenti italiani in balia dei disservizi"); va da sé che deve essere adottata ogni cautela per ridurre allo stretto necessario la banda impiegata.

Tornando al codice sopra illustrato, si può notare come, in aggiunta agli elementi `source`, siano presenti una serie di altri marcatori HTML che verranno visualizzati solo da quei browser che non riconosceranno l'elemento `<video>`. Questo ci consente di non lasciare a bocca asciutta chi utilizza un browser vecchio. Si propone un link per scaricare il file video nell'ipotesi che almeno il sistema operativo disponga di un lettore adeguato. All'interno del marcitore `<video>` è anche possibile riproporre i tag `<object>` ed `<embed>` per lasciare spazio a una consultazione del filmato anche sui vecchi browser.

### **LISTATO 5.11** Elemento `<video>` con sorgenti in formati diversi

```
<video controls>
...
<!-- subito prima della chiusura dell'elemento <video> si ricorre all'implementazione del video
vecchio stile -->
<object width="320" height="240" type="application/x-shockwave-flash">
<param name="movie" value="video.swf">
<param name="play" value="false">
<param name="allowfullscreen" value="true">
<embed width="320" height="240" src="video.swf">
```

```

allowfullscreen="true" type="application/x-shockwave-flash">
</embed>
</object>
<!-- il filmato può essere reso disponibile per il download mediante link ai formati diversi per tutti quei
browser che non supportano i tag object e embed --&gt;
&lt;/video&gt;
</pre>

```

La crescente diffusione del supporto per l'elemento `<video>` renderà progressivamente superflua quest'ultima parte. Tuttavia, i lati negativi di questa tecnica risiedono nell'obbligo di predisporre così tanti formati e soprattutto nel prevedere una generosa quantità di spazio su disco per poterli ospitare.

In sintesi, la Tabella 5.3 elenca i motivi che potrebbero spingervi a scegliere o dismettere questa nuova soluzione.

**TABELLA 5.3** Punti da valutare nell'uso dell'elemento `<video>`

ARGOMENTI A FAVORE	ARGOMENTI CONTRARI
Offre una maggiore interoperabilità con altre tecnologie web e può essere manipolato mediante interfaccia di programmazione (API) JavaScript, relativamente semplice da usare.	Supporto frammentato dei formati video (mp4, WebM e Ogg) a fronte dell'ubiquità di Flash. Mantenere i principali formati rappresenta un onere in termini di spazio su disco e tempo speso nella codifica del video.
Non espone a bachi e vulnerabilità di sicurezza, che vanno dal semplice crash del sistema a minacce più pericolose che esporrebbero il computer ad attacchi esterni come successo in passato per alcune tecnologie proprietarie.	Non garantisce quella protezione dei contenuti che, per esempio, Flash ha raggiunto con il suo protocollo RTMP (acronimo di Real-Time Messaging Protocol).
Migliora sia la semantica sia l'accessibilità della pagina web. Quest'ultima, a oggi, è disponibile solo attraverso l'introduzione di testo da sincronizzare con il video per mezzo delle API JavaScript (o anche grazie alla libreria Popcorn.js).	A oggi non è ancora possibile disporre di uno streaming video così evoluto che permetta di visualizzare con immediatezza e semplicità una parte arbitraria di un filmato come ormai si è abituati a fare.

## Popcorn.js mette il turbo al tuo video

Popcorn.js è una libreria JavaScript di facile impiego. Non bisogna essere un guru del linguaggio di scripting per sfruttarne le potenzialità. Grazie a questa libreria possiamo, per esempio, aggiungere dei sottotitoli al nostro video: curare l'accessibilità dei contenuti offerti non è solo una questione di "politicamente corretto" ma, più prosaicamente, è un modo saggio di ampliare la propria base di utenti garantendo una modalità alternativa di fruizione a soggetti che altrimenti ne sarebbero esclusi.

Per prima cosa scarichiamo la libreria da questo indirizzo: <http://mozilla.github.com/popcorn-js/>. Includiamo il file `popcorn.js` nella pagina web, come facciamo abitualmente con altre librerie esterne:

## LISTATO 5.12 Includiamo lo script nell'intestazione del documento

```
<head>
  [omissis]
  <script src="popcorn.js"></script>
  <script src="jquery.js"></script>
</head>
```

Popcorn necessita di JQuery per svolgere il suo compito: per questo motivo includiamo entrambe le librerie nel documento.

La relazione tra la libreria e il video cui vogliamo applicare le funzionalità aggiuntive (in questo caso i sottotitoli) è rappresentata dall'attributo `data-timeline-sources`.

## LISTATO 5.13 L'attributo `data-timeline-sources` determina la relazione tra libreria e risorsa video

```
<video src="video.ogv" data-timeline-sources="popcorn.xml"></video>
```

### Attributi `data*`

La specifica HTML5 (Paragrafo 3.2.3.8) stabilisce che l'autore del documento può fare ricorso ad attributi che abbiano il prefisso `data-` per introdurre informazioni che non è possibile contrassegnare in modo più adeguato. Questi dati non hanno effetto alcuno sulla stilizzazione della pagina web. I valori di questi attributi sono piuttosto da intendersi come dati grezzi messi a disposizione del linguaggio di scripting. Questo è proprio il caso dell'attributo `data-timeline-sources` utilizzato per fornire informazioni utili al funzionamento della libreria Popcorn.js.

I sottotitoli sono specificati attraverso il file `popcorn.xml` che segue una semplice sintassi.

## LISTATO 5.14 I sottotitoli sono applicati al video via XML

```
<popcorn>
  <subtitles language="it" align="center">
    <subtitle in="00:00:01:00" out="00:00:03:00">Sagome di
    legno...</subtitle>
    <subtitle in="00:00:04:00" out="00:00:06:00">...altre sagome di
    legno,</subtitle>
    <subtitle in="00:00:07:00" out="00:00:010:00">ma ci sono anche dei
    libri?</subtitle>
    <subtitle in="00:00:11:00" out="00:00:013:00">Eccoli!</subtitle>
  </subtitles>
</popcorn>
```

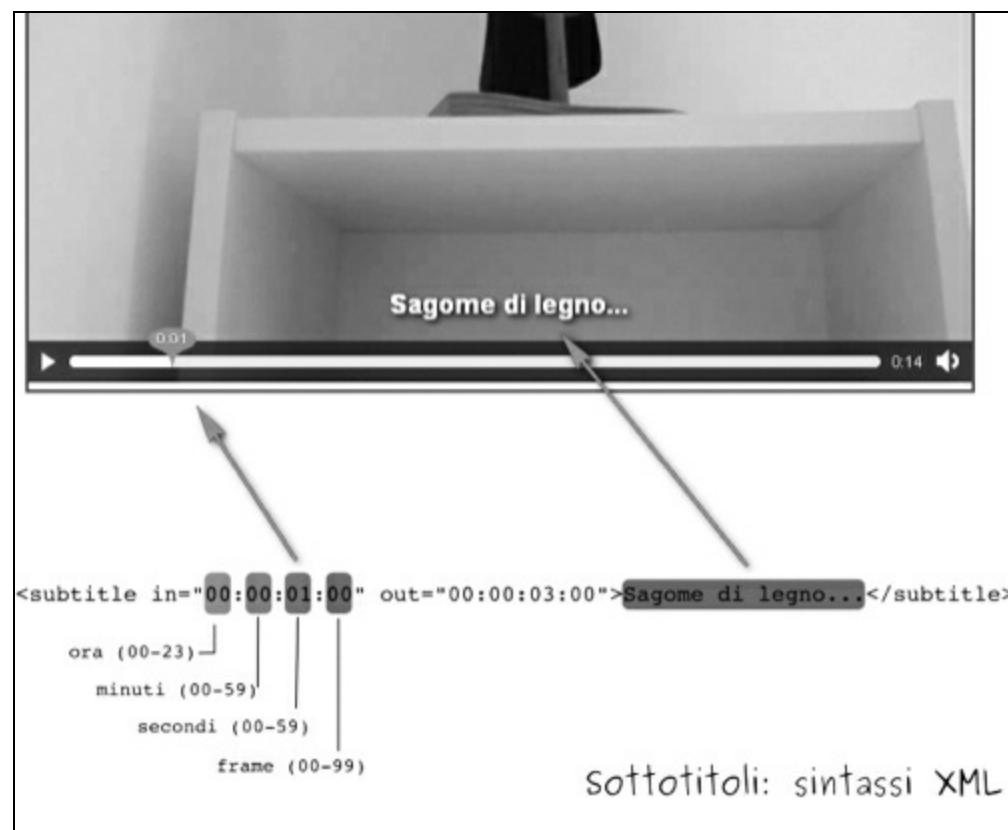
In questa configurazione il file è di facile lettura. All'interno di un tag `<popcorn>` che è alla radice di quest'albero XML, i singoli sottotitoli sono definiti nell'ambito del tag `<subtitles>`,

qui usato con due attributi:

- language: accetta la stringa di identificazione di una data lingua secondo lo standard ISO-639-1 (per cui "it" è usato per la lingua italiana, "es" per quella spagnola, "en" per quella inglese e così via);
- align: per allineare i sottotitoli a sinistra, centro o destra si useranno rispettivamente i valori `left`, `center` e `right`.

Per ogni sottotitolo si farà uso del tag `<subtitle>`, che presenta due diversi attributi `in` e `out` i quali accettano dati nello stesso formato: `ore:minuti:secondi:frames`. Il primo indica il momento dell'ingresso del sottotitolo; il grado di controllo esercitabile è molto elevato, potendo arrivare a determinare persino il frame per secondo di filmato. L'attributo `out` segna l'uscita di scena del sottotitolo.

L'immagine in Figura 5.7 illustra l'effetto di una dichiarazione di sottotitolo sul filmato qui illustrato con il player di Firefox 3.6.10.



**FIGURA 5.7** L'effetto prodotto da una dichiarazione XML del sottotitolo sul filmato.

In ultimo, per una corretta visualizzazione dei sottotitoli è necessario utilizzare i fogli di stile. Per questo motivo, sempre nell'intestazione del documento, aggiungiamo quanto segue:

**LISTATO 5.15** Fogli di stile applicati al div entro cui visualizzeremo il video

```

<style>
.video-div {
  border: 2px solid #554f33;
  width: 640px;
}
.video-div > div { padding-top: 185px; }
</style>

```

.video-div rappresenta il `<div>` nell'ambito del quale il filmato è inserito. In questo caso ci si limita a impostare un bordo e, cosa ancora più importante, una larghezza che è esattamente quella del video. Quest'ultima dichiarazione è utile perché fornisce il contesto entro cui opera il `<div>` in cui sono inseriti i sottotitoli.

La seconda regola ci aiuta a posizionare i sottotitoli nella parte bassa del filmato. Se non lo facessimo, verrebbe applicata la regola predefinita che centra i sottotitoli rispetto al filmato stesso e questo, il più delle volte, non è il risultato atteso. Mettendo assieme tutti i pezzi fin qui esaminati avremo:

### **LISTATO 5.16** Listato completo: i sottotitoli

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>La libreria di mio figlio: Video con sottotitoli</title>
<script src="popcorn.js"></script>
<script src="jquery.js"></script>
<style>
.video-div {
  border: 2px solid #554f33;
  width: 640px;
}
.video-div > div { padding-top: 185px; }
</style>
</head>
<body>
<div class="video-div">
<video width="640" height="480" data-timeline-
sources="xml/popcorn.xml" controls>
<source src="video.webm" type='video/webm; codecs="vp8, vorbis"'>
<source src="video.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
<source src="video.ogv" type='video/ogg; codecs="theora, vorbis"'>
<p>Scarica il video in formato <a href="video.webm">
title="Dimensioni: 853 Kb">WebM</a>,
<a href="video.mp4" title="Dimensioni: 4,98 MB">Mp4</a> o
<a href="video.ogv" title="Dimensioni: 3,34 MB">Ogv</a></p>

```

```
</video>
</div>
</body>
</html>
```

## L'elemento <audio>

<audio> e <video> condividono una serie di attributi che in entrambi gli elementi svolgono la stessa funzione e si comportano allo stesso modo. Si tratta di: `src`, `preload`, `autoplay`, `loop`, `controls`. In merito a quest'ultimo attributo, si ricorda che costituisce la sola modalità di rappresentazione grafica dell'elemento <audio>; non esplicitarlo comporta l'invisibilità dell'elemento, anche se resta possibile intervenire sull'elemento mediante il linguaggio di scripting.

Per quanto visto sopra in relazione a tali attributi potremmo dunque inserire un file audio nella nostra pagina web scrivendo:

### **LISTATO 5.17** Sintassi base per l'inserimento di un file audio

```
<audio src="audio.ogg" controls>
```

Tuttavia, anche il tag <audio> soffre, in questa fase, della stessa incertezza sul formato di codifica che in futuro prenderà il sopravvento. Si contendono lo scettro del mercato, da una parte un formato proprietario e arcinoto come l'mp3, e dall'altra il più oscuro e finora semisconosciuto codec audio Vorbis, racchiuso nel formato Ogg/Theora e per il momento confinato prevalentemente al mondo Linux.

Nella pratica quotidiana, ricalcando quanto sin qui visto per l'elemento <video> avremo un codice di questo tipo:

### **LISTATO 5.18** Elemento audio: un'ipotesi di implementazione

```
<audio controls>
  <source src="audio.ogg" type="audio/ogg; codecs=vorbis">
  <source src="audio.mp3" type="audio/mpeg">
  <object>
    <param name="src" value="audio.mp3">
    <param name="autoplay" value="false">
    <param name="controller" value="true">
    <embed type="audio/mpeg" src="audio.mp3" width="400" height="27"
      autoplay="false"></embed>
  </object>
  <p>Se non riesci ad eseguire il file audio direttamente dal browser
  puoi scaricarlo in formato <a href="audio.mp3">mp3</a>
```

```
o <a href="audio.ogg">ogg</a></p>
```

```
</audio>
```

## Le API JavaScript: il tuo telecomando!

Insieme ai tag `<video>` e `<audio>` si introducono metodi, eventi e attributi che rendono possibile interagire con gli elementi multimediali via JavaScript. Le potenziali applicazioni sono molte. Uno degli usi più frequenti consiste nel sostituire i controlli standard offerti dal browser con un set di comandi più ricco e con una stilizzazione personalizzata.

Di seguito si illustra un caso di utilizzo interattivo in cui il file audio può essere eseguito, oltre che attraverso l'ordinario tasto Play, anche a seguito della risposta esatta in un quiz (Figura 5.8).



The screenshot shows a BBC Learning English page. The main title is 'KEEP YOUR ENGLISH UP TO DATE'. On the left, there's a sidebar with links: 'Introduction', 'Make my day!', 'The full Monty!', 'FAQs', 'G.M.', 'Alcopops', 'Docusoaps', 'Text', 'Dis', 'e -', 'Mwah!', and 'Phwoar!'. The main content area features a photograph of a woman in a white coat standing in front of an ATM machine. Below the photo, the text 'Hole-in-the-wall' is mentioned. A 'Listen to Professor Crystal' button is present, with an audio player showing the file 'Hole-in-the-wall'. The BBC navigation bar at the top includes 'Home', 'News', 'Sport', 'Radio', 'TV', 'Weather', 'Languages', and 'BBC LEARNING ENGLISH'. A link 'Other series - go to index' is also visible.

**FIGURA 5.8** Per rendere più interattivo il materiale didattico si può lanciare l'esecuzione del file audio in funzione della risposta corretta data dall'utente.

In questa pagina web, disponibile nella sezione Learning English del sito della BBC, viene illustrato il significato della frase colloquiale "Hole In The Wall". Il materiale didattico messo a disposizione consiste in un file audio mp3 scaricabile, un file ram per sentire l'audio in streaming e il testo usato dal professor Crystal che spiega il significato del termine. La lezione potrebbe risultare più stimolante se venisse fornito all'utente anche un semplice quiz, per esempio una domanda a risposta multipla. In caso di risposta positiva l'utente è premiato con l'ascolto del file audio (la cui esecuzione può comunque avere luogo interagendo direttamente con i controlli dell'elemento sempre disponibili).

Ed ecco una implementazione plausibile.

```
<!DOCTYPE html>
<html>
  <head>
    [omissis]
  </head>
  <body>
    <article>
      <header>
        <h1>Aggiorna il tuo Inglese</h1>
      </header>
      <audio controls id="word">
        <source src="uptodate_holeinthewall.mp3">
        <source src="uptodate_holeinthewall.oga">
        <p>Scarica il file audio in formato <a href="uptodate_holeinthewall.mp3">MP3</a> o <a href="uptodate_holeinthewall.oga">OGA</a></p>
      </audio>
      <aside id="interactive">
        <form>
          <fieldset>
            <legend>Cosa significa "Hole-in-the-wall" in British English?</legend>
            <p><label for="cre"><input type="radio" name="definition" id="cre" value="crepa">Crepia (nel muro)</label></p>
            <p><label for="ban"><input type="radio" name="definition" id="ban" value="bancomat">Bancomat (colloquiale)</label></p>
            <p><label for="ent"><input type="radio" name="definition" name="app" id="ent" value="appartamento">Piccolo appartamento (colloquiale)</label></p>
          </fieldset>
        </form>
      [omissis]
      </aside>
      <p>... Segue trascrizione del file audio. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam cursus. Morbi ut mi. Nullam enim leo, egestas id, condimentum at, laoreet mattis, massa. Sed eleifend nonummy diam. Praesent mauris ante, elementum et, bibendum at, posuere sit amet, nibh.</p>
    </article>
  </body>
</html>
```

Abbiamo contrassegnato l'articolo racchiudendolo in un tag `<article>...</article>`. In apertura troviamo un titolo e l'elemento audio che qui propone due formati alternativi (mp3 e oga); il paragrafo inserito all'interno nella sezione `<audio>...</audio>` sarà visualizzato solo dai browser che non riconoscono il nuovo elemento. In questo caso si propone il download di entrambi i formati audio. Sempre all'interno dell'articolo abbiamo usato un elemento `<aside>` per delimitare un'area il cui contenuto è sì parte dell'articolo, ma lo

arricchisce in modo tale che se fosse omesso il contenuto principale non ne soffrirebbe. Questa è l'area nella quale abbiamo inserito il quiz a risposta multipla. Al termine di quest'area inizia il testo dell'articolo vero e proprio, che qui ho riportato solo con testo fittizio, il classico Lorem ipsum utilizzato come riempimento.

Analizziamo ora il contenuto dell'intestazione del documento.

#### **LISTATO 5.20** L'intestazione del documento

```
<head>
  <meta charset="utf-8">
  <title>Hole-In-The-Wall</title>
  <script>
    document.createElement("article");
    document.createElement("aside");
    document.createElement("header ");
  </script>
  <style>
    body { font-family: arial, sans-serif; }
    article { display: block; width: 70%; }
    aside { padding: 0.5em; float: right; width: 50%; }
  </style>
</head>
```

Quattro compiti vengono assolti nell'intestazione del documento.

- Si definisce esplicitamente la codifica dei caratteri a UTF-8: `<meta charset="utf-8">`; facendolo immediatamente all'apertura dell'intestazione ci proteggiamo da una potenziale falla di sicurezza (se non lo avete già fatto leggete, a tal proposito, il box "Il mistero dei 512 byte" nel Capitolo 2).
- Si imposta il titolo del documento: `<title>Hole-In-The-Wall</title>`.
- Lo script che crea in memoria un'occorrenza dei tag `<article>`, `<aside>` e `<header>` è necessario perché Internet Explorer nelle versioni precedenti alla 9 non è in grado di applicare altrimenti alcuno stile a questi marcatori. Si può fare a meno di questo script se si usa la libreria Modernizr, il cui impiego è caldamente consigliato! In tal caso al posto del seguente script:

#### **LISTATO 5.21** Lo script necessario per la corretta applicazione degli stili in IE < 9

```
<script>
  document.createElement("article");
  document.createElement("aside");
  document.createElement("header");
</script>
```

avremmo semplicemente scritto:

#### **LISTATO 5.22** Inclusione della libreria Modernizr, attualmente nella sua versione 1.6

```
<script src="modernizr-1.6.min.js"></script>
```

- In ultimo si definisce un set minimo di fogli di stile. Qui la sola cosa da notare è che definiamo esplicitamente l'articolo come un elemento di blocco con la regola `display: block`. In altri termini, chiediamo al browser che interpreti l'articolo come elemento che crea discontinuità prima e dopo il suo impiego. Ciò è utile per quei browser che non riconoscono l'elemento.

Nel Listato 5.19, il secondo omissis si riferisce a una parte dell'elemento `<aside>` precedentemente omessa. Qui inseriamo lo script in cui si applicano le logiche di verifica del supporto dell'elemento audio e, in caso positivo, il controllo della risposta esatta per determinare se possa essere scatenata l'esecuzione del file.

#### **LISTATO 5.23** Lo script di identificazione del supporto del tag `<audio>` e controllo della risposta esatta

```
<script>
  var AUDIO_APP = {};
  AUDIO_APP.canPlay = !!document.createElement("audio").canPlayType;
  if (AUDIO_APP.canPlay) {
    AUDIO_APP.audio = document.getElementsByTagName("audio")[0];
    AUDIO_APP.buttons = document.getElementsByTagName("input");
    AUDIO_APP.audio.addEventListener("play", function () {
      isPlaying = true;
    }, false);
    AUDIO_APP.audio.addEventListener("ended", function () {
      AUDIO_APP.isPlaying = false;
      uncheckRadioButtons();
    }, false);
  } else {
    document.getElementsByTagName("aside")[0].style.display = "none";
  }
  function checkAnswer(e) {
    if (AUDIO_APP.isPlaying) {
      return;
    }
    var elem = e.target || e.srcElement;
    if (elem.value == "bancomat") {
      AUDIO_APP.audio.play();
      AUDIO_APP.isPlaying = true;
    }
  }
</script>
```

```

    } else {
        alert("Errato! Ritenta :-)")
    }
}

for (var i = 0; i < AUDIO_APP.buttons.length; i++) {
    AUDIO_APP.buttons[i].onclick = checkAnswer;
}

function uncheckRadioButtons() {
    for (var i = 0; i < AUDIO_APP.buttons.length; i++) {
        if (AUDIO_APP.buttons[i].checked) {
            AUDIO_APP.buttons[i].checked = false;
        }
    }
}

</script>

```

Lo script ha inizio con la dichiarazione di un oggetto AUDIO\_APP cui ricondurre quelle che altrimenti sarebbero variabili globali e che saranno poi impiegate nelle diverse funzioni. Il primo obiettivo consiste nel definire se il browser supporta o meno l'elemento `<audio>` HTML5. Per far ciò conserviamo nella variabile `canPlay` la lettura del valore restituito dal metodo `canPlayType()` dell'oggetto audio. L'operatore `!!` serve per "tradurre" in booleano tale valore.

## Testare il supporto con Modernizr

Se avessimo utilizzato Modernizr avremmo potuto sostituire l'assegnazione seguente:

```
AUDIO_APP.canPlay = !!document.createElement("audio").canPlayType;
```

con la più sintetica ed elegante formula:

```
AUDIO_APP.canPlay = Modernizr.audio;
```

Se `AUDIO_APP.canPlay = true` ci troviamo di fronte a un browser in grado di interpretare correttamente l'elemento `<audio>`; non ci resta che predisporre lo script per accogliere la risposta dell'utente. A questo scopo recuperiamo prima il riferimento all'elemento audio, poi al set dei 3 radio-button, quindi stabiliamo che non appena il file audio è in esecuzione la variabile `AUDIO_APP.isPlaying` sia impostata a `true`, per converso la impostiamo a `false` quando viene scatenato l'evento `ended`, il che accade, com'è lecito attendersi, al termine dell'esecuzione del file audio. In questo caso rimuoviamo anche la risposta fornita dall'utente così da consentire una nuova iterazione.

Se il browser non supporta il nuovo elemento `<audio>`, `AUDIO_APP.canPlay` sarà uguale a `false`. In tal caso ci limitiamo a nascondere l'area interattiva racchiusa nel tag `<aside>`.

Quando l'utente seleziona una delle tre opzioni disponibili, fornendo una risposta, la

funzione `checkAnswer()` viene eseguita. Questa dapprima verifica se il file audio è in esecuzione; se lo è, la funzione ignora la risposta data dall'utente. In caso contrario si identifica l'elemento che ha scatenato l'evento.

## Assegnare un valore predefinito con l'operatore ||

Per identificare l'elemento che ha originato l'evento `onclick` si usa l'espressione seguente: `var elem = e.target || e.srcElement;` qui si noti l'uso dell'operatore `||` (doppio pipe) che, in questo contesto, serve per assegnare un valore predefinito alla variabile locale `elem`. Infatti se `e.target` risulta non essere definito, `e.srcElement` sarà eseguito e il suo risultato sarà assegnato a `elem`.

Se la risposta è corretta, parte l'esecuzione del file audio, altrimenti l'utente riceve un messaggio in finestra modale che lo invita a ritentare.

## Aggiorna il tuo Inglese



Segue trascrizione del file audio. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam cursus. Morbi ut mi. Nullam enim leo, egestas id, condimentum at, laoreet mattis, massa. Sed eleifend nonummy diam. Praesent mauris ante, elementum et, bibendum at, posuere sit amet, nibh.

Cosa significa "Hole-in-the-wall" in British English?

- Crepa (nel muro)
- Bancomat (colloquiale)
- Piccolo appartamento (colloquiale)

**FIGURA 5.9** Una riproposizione del contenuto didattico in chiave più interattiva sfruttando l'elemento audio HTML5 e le API JavaScript.

Altre, più originali, modalità di interazione sono alla nostra portata. Potreste sfruttare la possibilità di lanciare via JavaScript l'esecuzione contemporanea di più file audio per registrare lo stesso testo con voci diverse (una maschile, un'altra femminile) e passare istantaneamente dall'una versione all'altra a seconda delle preferenze dell'utente, per esempio con un clic del mouse su un pulsante d'opzione proprio come nell'esempio illustrato.

## Riferimenti alle risorse citate e altri link utili

- Marcatore `<video>` – la specifica HTML5: <http://www.w3.org/TR/html5/video.html#video>
- Marcatore `<audio>` – la specifica HTML5: <http://www.w3.org/TR/html5/video.html#audio>
- "Thoughts on Flash". Il titolo originale della lettera aperta di Steve Jobs su Adobe Flash: <http://www.apple.com/hotnews/thoughts-on-flash/>
- Lo scetticismo di Erik Huggers, responsabile dei sistemi informativi della TV

pubblica britannica BBC sul modello di sviluppo di HTML5, in un articolo del 13 agosto 2010: [http://www.bbc.co.uk/blogs/bbcinternet/2010/08/html5\\_open\\_standards\\_and\\_the\\_b.html](http://www.bbc.co.uk/blogs/bbcinternet/2010/08/html5_open_standards_and_the_b.html)

- Hulu, sito di video in streaming basato sul modello delle inserzioni pubblicitarie, spiega sul suo blog perché ritiene il video nativo HTML5 ancora non adatto al suo business: <http://blog.hulu.com/2010/05/13/pardon-our-dust/>
- Se il browser non riconosce il marcitore `<video>` la libreria JavaScript ricorre a Flash o Silverlight: <http://mediaelementjs.com/>
- Videojs è un'altra libreria JavaScript che si propone di risolvere i problemi di compatibilità con i vecchi browser; in più propone un'interfaccia di controllo video che rimane inalterata al cambiare di browser e sistemi operativi: <http://videojs.com/>
- PopCorn.js è una libreria JavaScript che consente di manipolare un file video, per esempio aggiungendo i sottotitoli come illustrato in questo capitolo: <http://mozilla.github.com/popcorn-js/>
- Per una lista completa dei codici di identificazione della lingua consultate l'indirizzo [http://en.wikipedia.org/wiki/List\\_of\\_ISO\\_639-1\\_codes](http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes) sempre utile nella remota, ma pur plausibile, ipotesi che abbiate bisogno di utilizzare sottotitoli in uzbeko.

## Conclusioni

Gli autori di pagine web si trovano oggi nella condizione di proporre file audio e video in modo più semplice di quanto fosse possibile in passato. Inoltre, grazie alle API JavaScript e alle librerie esterne, come dimostrato in questo capitolo, il controllo e la manipolazione di questi file compie un salto di qualità. È probabile che le prossime versioni del linguaggio facciano proprie in modo nativo le funzionalità aggiuntive che ora sono proposte da terze parti attraverso librerie JavaScript (si veda appunto il caso di Popcorn.js con i sottotitoli). Nel complesso è lecito aspettarsi una maggiore diffusione di risorse multimediali.

Occorre, in tale contesto evolutivo, ricordarsi di un'ovvia quanto trascurata realtà: non tutti gli utenti disporranno di connessioni affidabili e veloci, che sono un requisito fondamentale per una piacevole fruizione di contenuti multimediali. Questo è vero soprattutto se il sito si rivolge, per contenuti e lingua, al nostro paese, che soffre di un ritardo strutturale nel tasso di penetrazione della banda larga.

### ADSL: indietro tutta

L'ultimo rapporto Istat "Cittadini e nuove tecnologie" a oggi disponibile è quello del 2009. A pagina 5 si legge: "Considerando la percentuale di famiglie con almeno un componente tra i 16 e i 64 anni che possiede un accesso a Internet da casa, l'Italia è rimasta indietro rispetto a molti dei paesi della Comunità europea, risultando al ventunesimo posto, con un tasso di penetrazione del 53% rispetto alla media europea del 65%. Vicini all'Italia troviamo paesi come Cipro (53%) e Repubblica Ceca (54%), mentre Olanda, Svezia, Lussemburgo e Danimarca registrano un tasso di penetrazione che supera l'83%".

Poche cose sono più penose di un video che si vede a scatti o un file audio che riproduce la triste esperienza dell'ascolto di una qualunque stazione radiofonica mentre si è in viaggio sulla A10 (una delle autostrade italiane con il maggior numero di metri di galleria per chilometro di autostrada). Tenendo a mente questo problematico contesto operativo, sarà utile attenersi scrupolosamente ad alcune norme di buon senso:

- valutare la reale necessità di pre-caricare risorse multimediali senza ricorrere all'esplicito consenso del visitatore. Gli utenti di smartphone sono molto più sensibili di altri al consumo di banda per via dei piani tariffari che, quando non sono a tempo (spesso con scatti da 15 minuti!), sono appunto a traffico;
- proporre anche file multimediali di qualità più modesta e dunque, più leggeri;
- prediligere, laddove questo sia possibile, codec più efficienti per mettere a disposizione file più leggeri a parità di qualità offerta.

In relazione a quest'ultimo punto, sensibilizzare i propri visitatori, suggerendo di passare a browser che supportano codec più efficienti, è un modo sano di dare un contributo all'evoluzione del Web verso una situazione più sostenibile, curando al tempo stesso gli interessi dei propri utenti.

A proposito di browser, scrivo un'ultima annotazione prima di passare all'elemento `<canvas>`, altro "pezzo grosso" dell'HTML5. Internet Explorer 9 è il primo browser della Microsoft in grado di supportare questi nuovi componenti multimediali. Tuttavia preme qui far notare che questo browser non è compatibile con il sistema operativo XP, che ancora oggi rappresenta una parte largamente maggioritaria della base dei sistemi operativi facenti capo alla casa di Redmond. Anche in considerazione di ciò, date grande enfasi alle strategie di supporto ai browser più datati.

# Capitolo 6

## Canvas

L'elemento `<canvas>` può essere immaginato come una tela, una superficie rettangolare sulla quale disegnare. La prima cosa insolita, in questa metafora, è lo strumento con cui creare questi disegni: un linguaggio di scripting; la vostra matita è JavaScript, ma se volete darvi un tono davanti agli amici chiamate questa "matita" Canvas 2D API (se poi assumete un'aria seria e impostate un timbro vocale più profondo l'effetto scenico è assicurato). 2D sta per "2 Dimensioni", che poi rappresentano il contesto entro cui creare figure geometriche, grafici, testo, immagini e persino animazioni.

In questo capitolo utilizzeremo alcune semplici figure come pretesto per osservare da vicino il funzionamento dei metodi e delle proprietà che rappresentano gli strumenti di "scrittura" sul canvas. Non fatico a immaginare sia poco entusiasmante produrre righe e righe di codice per veder apparire figure geometriche ma, arrivando a padroneggiare lo strumento, si apre un universo di applicazioni tra le più disparate: si va da quelle più artistiche come dimostra la Figura 6.1, dove l'abilità di un programmatore è alla base di un'affascinante animazione (per saperne di più seguite il collegamento relativo nel paragrafo Riferimenti alle risorse citate e altri link utili), alla possibilità di produrre grafici aggiornati dinamicamente in base a flussi di dati provenienti dal server (si pensi per esempio a un diagramma a linee che riflette le fluttuazioni dell'indice della Borsa italiana FTSE MIB).



**FIGURA 6.1** Foglie e lucciole che si muovono al vento sotto un cielo stellato. Un uso artistico del `<canvas>`, il tutto in

L'argomento è ricco e lo spazio a nostra disposizione esiguo, motivo per cui iniziamo subito a familiarizzare con una parte importante della specifica HTML5 partendo dalla tabella che sintetizza il supporto dell'elemento presso i principali browser.

**TABELLA 6.1** Supporto dell'elemento <canvas>

ELEMENTO SUPPORTATO	BROWSER
<canvas>	Chrome 5*, Firefox 1.5, IE9, Opera 9, Safari 2+

\* Chrome sta sviluppando un supporto al Canvas 2D ancora più veloce attraverso un uso più esteso del processore grafico per accelerare la velocità di calcolo.

## <canvas>, primi passi

La sintassi base di un elemento <canvas> è minimalista.

**LISTATO 6.1** Sintassi base

```
<canvas></canvas>
```

Questa istruzione è di per sé sufficiente a inserire nel documento web una superficie larga 300 pixel e alta 150 pixel. 300×150, infatti, sono le dimensioni predefinite assegnate all'elemento se non le impostiamo esplicitamente usando gli attributi `width` (per la larghezza) e `height` (per l'altezza), gli unici due attributi specifici di questo elemento. Tutti gli altri attributi applicabili sono, come per l'attributo `id`, globali.

Ipotizzando di voler usare dimensioni diverse da quelle predefinite, per esempio 300×300, e avendo premura di mostrare un contenuto alternativo per quei browser o per quelle versioni di Internet Explorer che non riconoscono l'elemento, potremmo arricchire la precedente istruzione con qualcosa di appena più elaborato, come:

**LISTATO 6.2** Canvas quadrato da 300×300 pixel

```
<canvas id="tela" width="300" height="300">Il tuo browser non supporta l'elemento canvas</canvas>
```

Nel caso in cui l'elemento non sia supportato, il browser si limiterebbe a interpretare quanto inserito all'interno del marcatore stesso. Rifacendoci all'esempio sopra riportato, si vedrebbe la stringa di testo "Il tuo browser non supporta l'elemento canvas". È più corretto sostituire frasi simili con un vero contenuto alternativo per i browser più datati,

per esempio un'immagine del grafico che alternativamente sarebbe stato illustrato mediante l'elemento `<canvas>`.

## ExplorerCanvas

Google ha sviluppato una libreria il cui scopo è di "abilitare" il riconoscimento dell'elemento `<canvas>` in Internet Explorer. Il nome di questa libreria è appunto ExplorerCanvas (è scaricabile dal sito <http://code.google.com/p/explorercanvas>).

Essa rende accessibile una parte delle funzionalità previste dalle API JavaScript. La libreria viene inclusa come qualunque altra libreria JavaScript, tuttavia, poiché il suo scopo è di colmare una lacuna specifica dei browser di casa Redmond, sarà preferibile fare uso di commenti condizionali, ossia di commenti che saranno interpretati come tali da tutti i browser a eccezione di Internet Explorer, che è in grado di interpretare il costrutto `if` per stabilire se interpretare o meno quanto in essi racchiuso. Possiamo adottare questa soluzione:

```
<!-- [if lt IE 9]><script src="excanvas.js"></script><![endif]-->
```

che si legge "Se il browser è Internet Explorer in una qualunque versione inferiore alla 9, procedi al caricamento della risorsa esterna, ossia il file `excanvas.js`". In questo modo evitiamo che browser diversi da Explorer assumano l'onere di interpretare una risorsa di cui non avranno necessità

D'altro canto, se volessimo visualizzare una pagina web contenente solo questo elemento, non vedremmo altro che una pagina bianca. La superficie sulla quale disegnare è ancora immacolata, spetta a noi scriverci usando JavaScript. La prima mossa è ottenere un riferimento all'elemento e testarne il supporto.

### **LISTATO 6.3** Test di supporto dell'elemento `<canvas>` con Modernizr

```
<script>
var canvas = document.getElementById("tela");
if (Modernizr.canvas) {
  canvas.getContext("2d");
  // si scrive sul canvas
}
</script>
```

Si inizia con l'ottenere un riferimento all'oggetto che lo rappresenta. Facciamo uso dell'attributo `id` per avere questo riferimento attraverso il metodo `getElementById()`. Ora non rimane che verificare se l'elemento è supportato. Se stiamo utilizzando Modernizr, il test si risolve nel controllare che la proprietà booleana `canvas` dell'oggetto `Modernizr` restituisca `true`.

Volendo fare a meno di Modernizr, possiamo testare direttamente il supporto dell'elemento leggendo il risultato restituito dal metodo `getContext()`. Del resto questo è esattamente quello che Modernizr fa per noi quando leggiamo il valore della proprietà booleana `canvas`.

### **LISTATO 6.4** Test di supporto "fai da te"

```
<script>  
if (!document.createElement("canvas").getContext) {  
    // canvas supportato  
} else {  
    // mancato supporto. Si usa excanvas.js per Explorer  
    // in alternativa si propone un contenuto statico.  
}  
</script>
```

Se questo è il caso, stiamo utilizzando un browser che supporta l'elemento, quindi possiamo ottenere un riferimento a quel contesto bidimensionale nell'ambito del quale operare. Per fare ciò utilizziamo il metodo `getContext()` che, a oggi, accetta solo il valore `"2d"`.

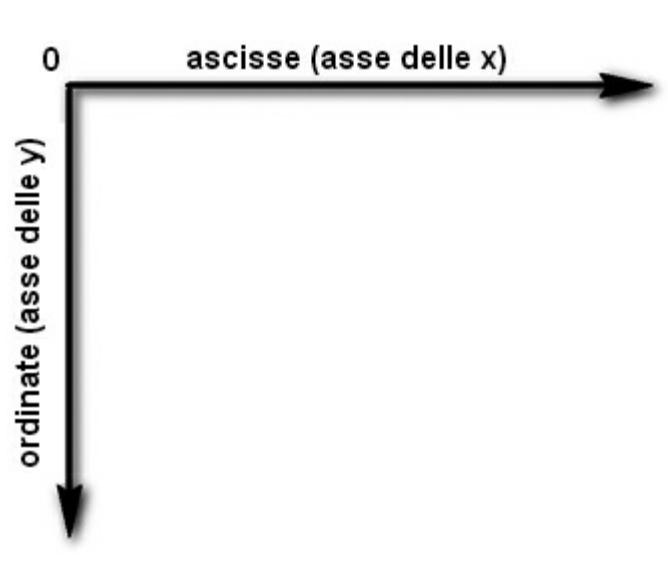
## Contesto tridimensionale

Solo fino a pochi anni fa l'esperienza delle tre dimensioni su Web era una chimera, ma nel volgere di poco tempo la convergente evoluzione di elementi diversi (diffusione della banda larga, browser in grado di sfruttare l'acceleratore hardware e tecnologie web 3D native, ossia che il browser sa interpretare senza l'ausilio di componenti esterni) ha fatto sì che il contesto tridimensionale sia ormai alla nostra portata.

Attualmente è in fase sperimentale un contesto tridimensionale chiamato WebGL, che nasce da un'intensa attività del gruppo di sviluppatori facenti capo a Mozilla Foundation. L'interesse su questa tecnologia si è rapidamente esteso tanto da coinvolgere esponenti dei principali browser oggi in commercio, che attualmente lavorano insieme alla stesura di una specifica stabile nell'ambito del consorzio Khronos Group.

È arrivato il momento di familiarizzare con questo famigerato "contesto bidimensionale", un altro modo in cui possiamo chiamare quella superficie rettangolare (o nel nostro caso quadrata) su cui possiamo disegnare. Sin dalla scuola secondaria di primo grado (la vecchia "scuola media") dovreste avere avuto dimestichezza con il piano cartesiano (Figura 6.2). Già immagino i primi prepotenti sbadigli farsi largo e quegli occhietti ormai ridotti a uno spillo che si chiudono senza appello: invece è proprio questo il momento di mostrare un minimo di dedizione alla causa della Conoscenza.

Prometto di essere breve, nel tentativo di non estinguere tanto ardore per HTML5. Voi promettetemi un barlume di attenzione. Nel piano bidimensionale vi sono i seguenti elementi.



**FIGURA 6.2** Un piano bidimensionale nei suoi tre elementi: origine, ascissa e ordinata.

- L'asse delle ascisse, o asse delle x: è l'asse orizzontale, riferendoci alla Figura 6.2. Il valore misurato lungo tale asse aumenta a mano a mano che si allontana dall'origine e spostandosi verso destra.
- L'asse delle ordinate, o asse delle y: è il nome dell'asse verticale. Il valore misurato lungo di esso aumenta tanto più ci si sposta verso il basso.
- L'origine: il punto in corrispondenza del quale i due assi si intersecano. Tutti gli elementi che appaiono all'interno del `<canvas>` sono disposti in relazione all'origine, le cui coordinate sono (0, 0).

## Un esempio pratico

Con queste informazioni in nostro possesso diventa più semplice capire come funzionano i metodi necessari per disegnare figure geometriche. Di seguito, ecco il codice che produce l'immagine illustrata in Figura 6.3. Il bordo più esterno è applicato via CSS con una dichiarazione del tipo: `canvas {border: 1px solid #666}`. Sebbene l'esempio non abbia stravolgenti applicazioni pratiche, si presta bene per introdurre le funzioni dei principali metodi nell'ambito del Canvas 2D API.



**FIGURA 6.3** Sulla sinistra il "segno di spunta" come appare su Chrome 6, sulla destra il risultato su Internet Explorer 6 (grazie a excanvas.js).

## LISTATO 6.5 Segno di spunta su box verde scurofunction canvasJob() {

```
var canvas = document.getElementById("tela");
var context = null;
if (Modernizr.canvas) {
    context = canvas.getContext("2d");
    context.fillStyle = "rgb(18, 83, 38)";
    context.fillRect(30,30,100,100);
    context.fillStyle = "rgb(30, 70, 23)";
    context.strokeRect(29,29,102,102);
    context.beginPath();
    context.moveTo(60,60);
    context.lineTo(100,120);
    context.moveTo(100,120);
    context.lineTo(120,20)
    context.lineWidth = 15;
    context.lineCap = "round";
    context.strokeStyle = "rgba(32, 345, 64, 0.5)";
    context.stroke();
}
}
</script>
```

Vediamo nel dettaglio come si ottiene il risultato illustrato in Figura 6.3. Abbiamo sin qui visto come sia necessario acquisire un riferimento al contesto dell'elemento `<canvas>`.

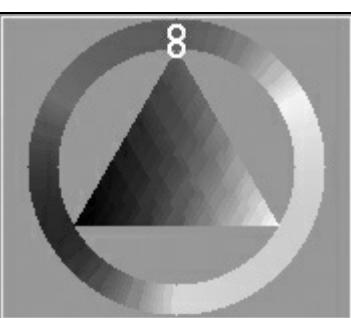
Ogni oggetto viene scritto sfruttando i suoi metodi. Il colore predefinito è il nero ma, volendo optare per un più rilassante verde da tavolo da gioco, scriviamo:

## LISTATO 6.6 fillStyle: utile per impostare un colore diverso da quello predefinito (nero)

```
context.fillStyle = "rgb(18, 83, 38)";
```

È possibile definire un colore applicando le stesse regole già note per i fogli di stile; questo porta ad avere ben sette diverse opzioni disponibili per la scelta del colore.

- L'operatore `rgb()`, come nell'esempio sopra riportato, che accetta 3 numeri da 0 a 255 per ciascun colore.
- L'operatore `rgb()` che, a differenza di quanto sopra, impiega le percentuali al posto dei numeri.
- L'operatore `rgba()`, introdotto con CSS3, che aggiunge al colore anche informazioni sull'opacità. Nel codice di esempio si sfrutta proprio questo operatore per sovrapporre due elementi con un effetto di trasparenza. Accetta valori tra 0 e 1, estremi compresi.



**FIGURA 6.4** Strumento per la scelta del colore secondo il modello HSL estrapolato da Gimp.

- L'operatore `hsl()`, introdotto con CSS3. Nell'acronimo `hsl`, `h` sta per hue, ossia sfumatura. Questa è definita per mezzo di un numero nell'intervallo da 0 a 360; il numero corrisponde a un angolo, misurato in gradi, che permette di identificare un dato colore in base alla sua disposizione lungo questa circonferenza (ogni programma di manipolazione delle immagini ne presenta una; in Figura 6.4 si illustra lo strumento come proposto dal software di elaborazione grafica Gimp). `s` sta invece per saturation (saturazione); accetta un valore percentuale. Tanto più questo numero tende a zero, tanto meno saturo è il colore. `l`, infine, sta per lightness (luminosità); anch'essa accetta valori percentuali.
- L'operatore `hsla()`, introdotto con CSS3, aggiunge all'operatore `hsl()` informazioni relative all'opacità del colore. Accetta valori tra 0 e 1, estremi compresi.
- La notazione composta da `#` seguito da sei cifre esadecimali, per cui si utilizzano due cifre esadecimali per ciascun colore.
- Una forma contratta della notazione precedente, composta da `#` seguito da tre cifre esadecimali, per cui si utilizza un solo esadecimale per ciascun colore.

#### **LISTATO 6.7** Definizione del rettangolo

```
context.fillRect(30,30,100,100);
```

`fillRect(x, y, larghezza, altezza)` è il metodo attraverso il quale definiamo posizione e dimensioni del rettangolo. Nell'esempio abbiamo creato un quadrato di lato 100 pixel posto a 30 pixel a destra e in basso rispetto all'origine.

Cambiamo di nuovo colore utilizzando un verde più scuro per disegnare un bordo attorno a questo quadrato. Per farlo invochiamo un altro metodo che disegna solo il bordo, dunque il perimetro, di un rettangolo.

#### **LISTATO 6.8** Disegnare il perimetro di un rettangolo con `strokeRect()`

```
context.strokeRect(29,29,102,102);
```

Gli argomenti del metodo `strokeRect()` seguono le stesse regole già viste per il metodo

`fillRect()`: i primi due argomenti sono le coordinate dell'angolo più in alto a sinistra del rettangolo; il terzo e il quarto argomento permettono di impostare rispettivamente larghezza e altezza del rettangolo. Mentre `fillRect()` usa il colore per riempire tutto il rettangolo, il metodo `strokeRect()` impiega il colore per tratteggiarne solo il perimetro.

## clearRect()

Pur se non presente nel listato, per completezza si cita qui `clearRect(x, y, larghezza, altezza)`, metodo che opera anch'esso sulle forme rettangolari e come `fillRect()` e `strokeRect()` accetta gli stessi parametri. Lo scopo di questo metodo è di ripulire, rendendo dunque trasparente, l'area rettangolare identificata attraverso le coordinate e le dimensioni definite per mezzo dei suoi quattro argomenti.

Per costruire il “segno di spunta” dobbiamo tracciare due linee oblique. Ed ecco il codice strettamente necessario per raggiungere l’obiettivo:

### **LISTATO 6.9** Scrittura del “segno di spunta”

```
context.beginPath();
context.moveTo(60, 60);
context.lineTo(100, 120);
context.moveTo(100, 120);
context.lineTo(120, 20)
```

Si segnala la creazione di una nuova figura con il metodo `beginPath()`. A questo punto si usa il metodo `moveTo(x, y)` che, come suggerisce il nome, consente di spostarsi in un punto preciso del `<canvas>`. Nel nostro caso esso è un punto a 60 pixel a destra e in basso rispetto all’origine. Il successivo metodo `lineTo(x, y)` fa sì che venga tracciata una riga dal punto di coordinate `(60, 60)` raggiunto in precedenza fino al nuovo punto identificato dalle coordinate `(100, 120)`. Per tracciare la seconda linea non dobbiamo fare niente di diverso: ci spostiamo nel nuovo punto di coordinate `(100, 120)` per mezzo del metodo `moveTo()` e da qui tracciamo una linea fino a un punto di coordinate `(120, 20)` posto all'esterno dell'area del quadrato.

Dopo aver definito il tracciato che la linea dovrà compiere per rappresentare il segno di spunta, ci dedichiamo ad alcuni ulteriori dettagli, quali spessore, formato e colore.

Per lo spessore si ricorre alla proprietà `lineWidth` dell’oggetto `context`.

### **LISTATO 6.10** Definizione dello spessore di una linea

```
context.lineWidth = 15;
```

Ogni tentativo di attribuire a questa proprietà un valore nullo o negativo sarà ignorato. Per quel che attiene al formato delle estremità della linea, si può intervenire impostando

uno dei tre possibili valori previsti per la proprietà `lineCap`.

- `butt`. Questo è il valore predefinito. La linea termina con un'estremità perpendicolare alla sua direzione.
- `round`. Un semi-cerchio è posto alle estremità della linea.
- `square`. Un rettangolo con altezza pari alla larghezza della linea e larghezza pari a metà della larghezza della linea è poggiato alle estremità della riga (al di là dello scioglilingua, c'è da chiedersi perché mai a questa "configurazione" non sia stato dato il nome "rect").

Volendo applicare al "segno di spunta" fin qui disegnato un colore diverso dal nero (il colore predefinito), impostiamo un valore opportuno per la proprietà `strokeStyle`.

#### **LISTATO 6.11** Operatore `gba()`: come impostare il colore e l'opacità

```
context.strokeStyle = 'rgba(32, 345, 64, 0.5)';
```

Avendo impostato un valore intermedio per l'opacità (`0.5`) vedremo il box verde anche attraverso il segno di spunta. Per visualizzare nel `<canvas>` questi tratti finalizziamo le operazioni di scrittura con il metodo `stroke()`.

#### **LISTATO 6.12** `stroke()`: il finalizzatore

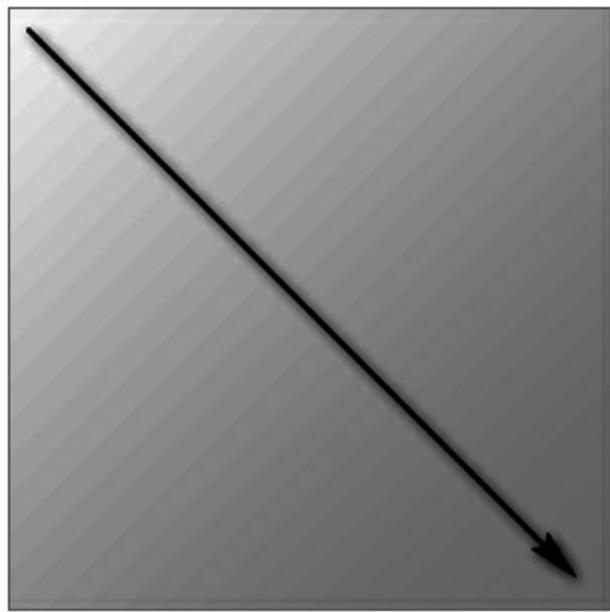
```
context.stroke();
```

Il metodo `stroke()` esegue in concreto l'operazione di scrittura dei tratti preventivamente impostati; in altri termini, applica il disegno secondo le regole definite in precedenza. In alternativa, il metodo `fill()` svolge una funzione analoga. La differenza tra i due metodi è che il primo agisce sul bordo, sul perimetro della figura, mentre il secondo ne colora l'area.

## Gradienti lineari e radienti

Abbiamo visto come applicare i colori per mezzo dei metodi `strokeStyle()` e `fillStyle()`. Si tratta, in questo caso, di uno stesso colore applicato uniformemente al perimetro o all'intera area della figura geometrica. Esiste tuttavia la possibilità di specificare un gradiente, che nella sua rappresentazione più semplice non è altro che una transizione lineare da un colore a un altro. Il codice che segue propone una transizione lineare dal giallo al rosso, come visualizzato dall'immagine in Figura 6.5.

(0,0) origine del canvas



(300,300)

**FIGURA 6.5** Transizione lineare tra giallo e rosso.

**LISTATO 6.13** Gradiente lineare tra 2 colori

```
function canvasJob() {  
  var context = document.getElementById("tela").getContext("2d");  
  var gradient = context.createLinearGradient(0, 0, 300, 300);  
  gradient.addColorStop(0, "#ff0");  
  gradient.addColorStop(1, "#f00");  
  context.fillStyle = gradient;  
  context.fillRect(0,0,300,300);  
}
```

Analizziamo il codice. La prima dichiarazione della funzione `canvasJob()` ha lo scopo di acquisire un riferimento al contesto entro il quale operare. Sarebbe opportuno adottare Modernizr o, in alternativa, la tecnica fai-da-te di cui si è scritto nel Listato 6.5 per capire se l'elemento è effettivamente supportato, ma in questa e nelle funzioni che seguiranno si omette questa parte per motivi di spazio.

Per creare un gradiente lineare abbiamo bisogno di tre tipi di informazioni:

- la direzione verso cui ha luogo la transizione tra due o più colori;
- i colori coinvolti in questa transizione;
- l'avvicendamento tra i colori del gradiente;
- l'applicazione del gradiente al metodo che inserisce in concreto il colore: per quanto visto sopra può trattarsi tanto del metodo `fillStyle()` quanto del metodo `strokeStyle()`.

## LISTATO 6.14 Un gradiente lineare

```
var gradient = context.createLinearGradient(0, 0, 300, 300);
```

Il metodo `createLinearGradient()` accetta quattro argomenti. I primi due rappresentano le coordinate del punto di partenza del gradiente che, nel caso in oggetto, identificano anche l'origine del canvas. Il terzo e il quarto argomento sono rispettivamente ascissa e ordinata del punto di destinazione del gradiente lineare. Questo verso è enfatizzato per mezzo della freccia nera presente in Figura 6.5. Il metodo, inoltre, restituisce un riferimento al gradiente cui ora possiamo aggiungere le informazioni relative ai colori.

## LISTATO 6.15 Colori e transizione del gradiente lineare

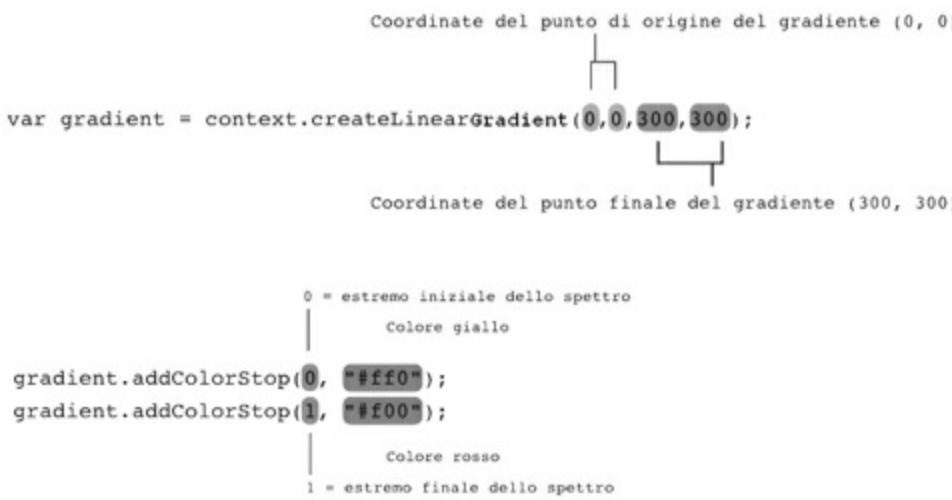
```
gradient.addColorStop(0, "#ff0");
gradient.addColorStop(1, "#f00");
```

`addColorStop()` è il metodo attraverso il quale si stabilisce numero di colori e loro modalità di successione nel gradiente.

Il primo argomento identifica la posizione del colore all'interno del gradiente: `0` significa che il colore è posto all'inizio del gradiente, mentre `1` significa che è posto al termine del gradiente stesso.

Il secondo parametro identifica il colore, qui impostato per mezzo di una tripletta esadecimale (`#ff0` = giallo, mentre `#f00` = rosso). La Figura 6.6 sintetizza le informazioni salienti.

### Gradiente lineare: transizione tra 2 colori

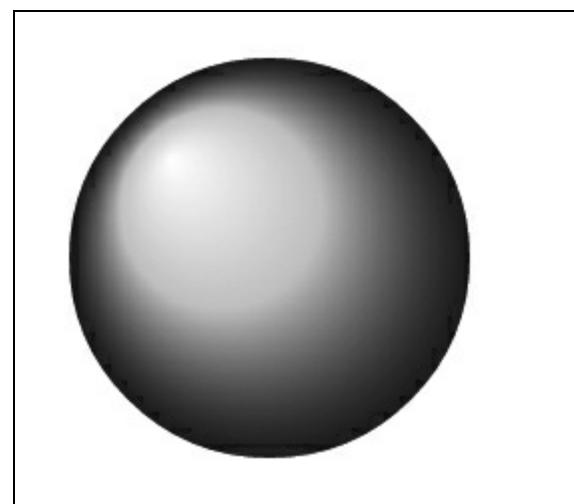


**FIGURA 6.6** Il codice di configurazione di un gradiente lineare.

Ne risulta che il gradiente da noi impostato inizia dal giallo e digrada verso il rosso man-

mano che si sposta dal punto iniziale di coordinate  $(0, 0)$  al punto finale di coordinate  $(300, 300)$ .

In aggiunta al gradiente lineare esiste anche il gradiente di tipo radiante per via del metodo `createRadialGradient(x0, y0, r0, x1, y1, r1)`. In questo caso il gradiente non si sviluppa tra due punti, come visto in precedenza, bensì tra due cerchi: il primo ha un'origine di coordinate `x0` e `y0`, mentre `r0` rappresenta il raggio. In modo analogo, `x1` e `y1` sono le coordinate del secondo cerchio e `r1` ne costituisce il raggio. Si può dunque immaginare che la sfumatura tra i colori abbia luogo nell'ambito di una figura cilindrica o conica che si sviluppa tra i due cerchi. Tale gradiente si può usare per produrre effetti come quello illustrato in Figura 6.7.



**FIGURA 6.7** Il gradiente di tipo radiante qui è sfruttato per dare l'illusione di stare osservando una sfera.

Ecco di seguito il codice impiegato per ottenere questo risultato:

**LISTATO 6.16** Gradiente di tipo radiante composto da 3 colori applicato a un cerchio

```
function canvasJob() {  
  var context = document.getElementById("tela").getContext("2d");  
  var gradient = context.createRadialGradient(300, 100, 0, 350, 150, 100);  
  gradient.addColorStop(0, "#fff");  
  gradient.addColorStop(0.5, "#ccc");  
  gradient.addColorStop(1, "#000");  
  
  context.fillStyle = gradient;  
  
  context.beginPath();  
  context.arc(350, 150, 100, 0, Math.PI * 2, false);  
  context.fill();  
}
```

Le novità introdotte rispetto al listato precedente sono tre. Alla prima abbiamo già fatto cenno: si usa un gradiente non lineare ma di tipo radiante. La seconda novità riguarda il

numero di colori che si alternano in questa sfumatura. Se prima erano solo due, qui ne abbiamo impiegati tre: la sfumatura ha inizio con il bianco (#fff) (il primo argomento uguale a zero sta proprio a indicare l'origine del gradiente), a seguire vi è un grigio (#ccc), quindi si termina con il nero (#000).

L'altra novità risiede nella figura geometrica cui abbiamo applicato tale gradiente. Per enfatizzarne l'impiego, e per illustrare ancora un'altra piccola parte del Canvas API 2D, si è qui ricorso a un cerchio, disegnato per mezzo del metodo `arc()` di cui analizziamo gli argomenti in dettaglio.

**LISTATO 6.17** Il significato degli argomenti del metodo `arc()`

```
arc(x, y, lunghezza_raggio, angolo_iniziale_in_radiani, angolo_finale_in_radiani, senso_antiorario)
```

- I primi due argomenti contrassegnati con le lettere  $(x, y)$  rappresentano le coordinate del punto di origine del cerchio.
- Il terzo parametro esplicita la lunghezza del raggio di questo cerchio.
- Il quarto e il quinto argomento identificano due punti posti lungo la circonferenza del cerchio: il primo è il punto dal quale si inizia a disegnare l'arco, il secondo è il punto di arrivo della figura. Per tali argomenti il valore è espresso in radienti. Con sommo gaudio degli amanti della trigonometria, riporto la definizione del radiente come pubblicata da Wikipedia: "Il radiente rappresenta il rapporto tra la lunghezza di un arco di circonferenza spezzato dall'angolo, e la lunghezza del raggio di tale circonferenza". Chiaro, no? In altri termini, dato un angolo espresso in gradi, si ottiene l'angolo espresso in radienti applicando la seguente formula:  $(\text{gradi} \cdot \pi)/180$ . In Tabella 6.2 ho riportato alcuni valori espressi in gradi e in radienti.
- L'ultimo argomento stabilisce quale sia il verso da applicare alla scrittura della figura. L'argomento è un booleano, che dunque accetta il valore `false` per indicare una scrittura in senso orario e `true` per un senso antiorario.

**TABELLA 6.2** Da gradi a radienti: un esempio di conversione

GRADI	FORMULA PER PASSARE AI RADIANTI	RADIANTI
0	$0 \cdot \pi/180$	0
45	$45 \cdot \pi/180$	0,785398163397448
90	$90 \cdot \pi/180$	1,5707963267949
180	$180 \cdot \pi/180$	3,14159265358979
360	$360 \cdot \pi/180$	6,28318530717959

## NOTA

In JavaScript, anziché riportare esplicitamente il valore di pi greco per poi arrestarsi a un numero arbitrario di cifre decimali, è opportuno usare la costante `Math.PI` come fatto nel codice sopra riportato.

## Immagini

L'elemento `<canvas>` consente anche di rappresentare immagini al suo interno. Per ottenere un tale risultato è possibile impiegare il metodo `drawImage()`.

### **LISTATO 6.18** Visualizzazione di un'immagine all'interno dell'elemento `<canvas>`

```
<script>
  var img=document.createElement("img");
  img.setAttribute("src", "jump.jpg");
  function canvasImg() {
    var context =
      document.getElementById("jump_canvas").getContext("2d");
    context.drawImage(img, 0, 0);
  }
</script>
```

Le prime due righe dello script servono per creare un'immagine aggiungendo al DOM (Document Object Model, ossia modello del documento a oggetti) un elemento `<img>` e specificandone di seguito l'attributo `src` con il relativo percorso sul server rispetto alla pagina che ospita l'immagine (altrimenti detto percorso relativo): in tal caso l'immagine e il file HTML risiedono nella stessa directory. In alternativa, avremmo potuto creare l'immagine sfruttando l'oggetto JavaScript `Image` operando come segue:

### **LISTATO 6.19** Creare un'immagine via JavaScript: un'alternativa

```
var img = new Image();
img.src = "jump.jpg";
```

Se l'immagine è già presente nel documento HTML, è possibile recuperarne un riferimento come facciamo abitualmente utilizzando il metodo `getElementById()`. All'interno della funzione `canvasImg()` si ottiene dapprima un riferimento all'oggetto `context`, quindi si inserisce l'immagine all'interno del contesto. Esaminiamo nel dettaglio il metodo `drawImage()` che rende ciò possibile.

## drawImage(image, dx, dy);

drawImage() accetta tre argomenti. Il primo è un riferimento all'immagine che si intende inserire nell'elemento `<canvas>`. Abbiamo visto che esistono diversi modi per ottenere questo riferimento. Il secondo e il terzo elemento sono le coordinate in cui verrà a trovarsi il vertice dell'immagine posto in alto e a destra. Poiché nel codice di esempio le coordinate sono entrambe pari a zero, significa che questo vertice coinciderà con l'origine del `<canvas>`. Per rendere evidente il modo in cui opera questo metodo, si attribuisce un bordo all'elemento `<canvas>` attraverso i fogli di stile e si impostano coordinate diverse da zero. Per esempio, ipotizziamo di modificare il metodo `drawImage()` nel modo seguente:

### LISTATO 6.20 drawImage(image, dx, dy)

```
context.drawImage(img, 60, 90);
```

Il risultato è apprezzabile in Figura 6.8.



**FIGURA 6.8** Il secondo e il terzo argomento sono le coordinate del vertice della foto in alto a sinistra.

Lo stesso metodo può essere invocato con altri due differenti set di parametri.

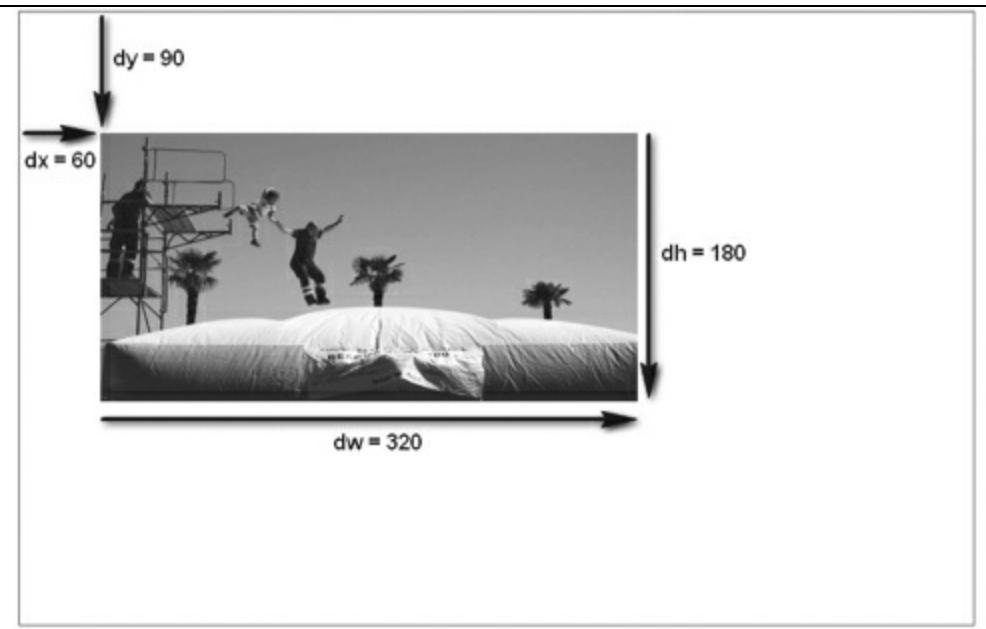
```
drawImage(image, dx, dy, dw, dh);
```

I primi tre argomenti ci sono noti. A questi se ne aggiungono altri due, che ci permettono di intervenire sull'immagine ridimensionandola. L'immagine `jump.jpg` è larga 640 pixel e lunga 361. Ipotizzando di volerla ridurre della metà si scriverà:

## LISTATO 6.21 `drawImage(image, dx, dy, dw, dh);`

```
context.drawImage(image, 60, 90, 320, 180);
```

Così facendo si otterrà il risultato visibile in Figura 6.9.



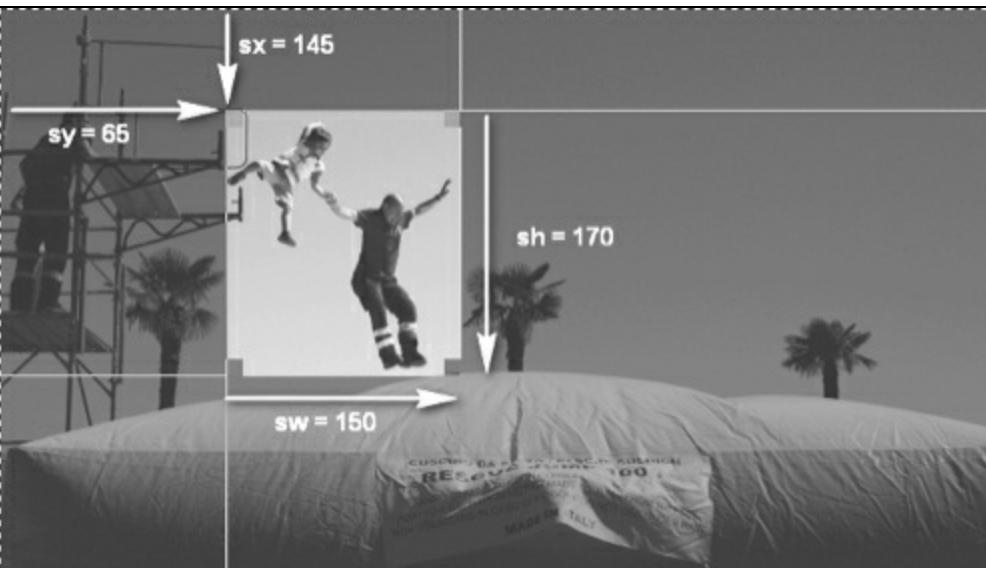
**FIGURA 6.9** L'immagine risulta più piccola rispetto alle dimensioni originali.

```
drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh);
```

Esiste una terza opzione. In questo caso i parametri sono ben nove. Ai precedenti, sin qui discussi, se ne aggiungono altri quattro (sx, sy, sw, sh). Grazie a questo set aggiuntivo di argomenti siamo in grado di identificare un'area rettangolare all'interno dell'immagine

`jump.jpg`.

Supponiamo, per esempio, di voler estrarre dalla foto solo il vigile del fuoco e il bambino in volo per poi ingrandire quest'area della foto. In questo caso prendiamo le coordinate dell'area della foto che li interessa. Come evidente dalla Figura 6.10, i punti `sx` e `sy`, pari rispettivamente a 145 e 65 pixel, sono le coordinate del vertice posto più in alto a sinistra di questo ipotetico rettangolo che avrà lunghezza pari a 150 e altezza pari a 170. Adesso sappiamo già come usare gli altri argomenti. Utilizziamo `dx` e `dy` per stabilire dove quest'immagine sarà posta nell'ambito del `<canvas>` e con `dw` e `dh` ne definiamo le dimensioni. Se pensiamo di voler applicare uno zoom  $\times 2$ , non dovremo far altro che raddoppiare i valori assunti da `sw` e `sh` così che `dw = sw * 2` e `dh = sh * 2`.

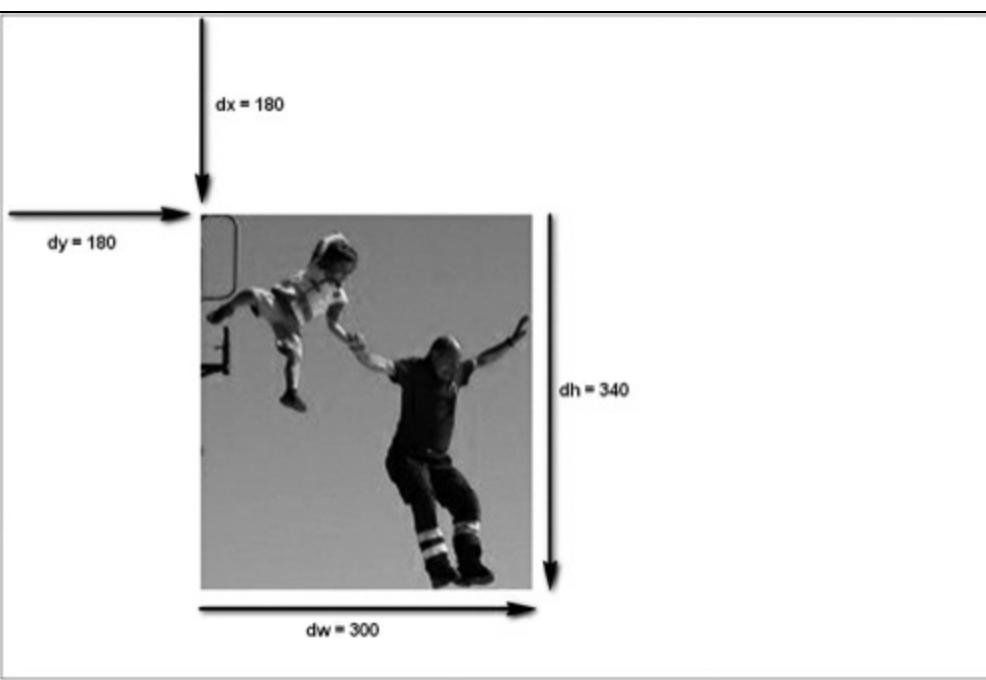


**FIGURA 6.10** Si identifica l'area della foto su cui si vuole operare.

**LISTATO 6.22** `drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh);`

```
context.drawImage(img, 145, 65, 150, 170, 180, 180, 300, 340);
```

Il risultato è apprezzabile in Figura 6.11.



**FIGURA 6.11** Una porzione dell'immagine è stata estrapolata dal contesto, spostata in un'area diversa del canvas e su di essa è stata applicato uno zoom x2.

Una volta che l'immagine è acquisita all'interno del `<canvas>`, può essere oggetto di manipolazione sia per riprodurre funzionalità analoghe a quelle esistenti nei software di foto ritocco (per esempio la rimozione degli occhi rossi, lo zoom su aree dell'immagine al passaggio del mouse ecc.), sia per realizzare composizioni artistiche come quelle applicabili con la libreria, scritta da David De Sandro, "Close-pixelate". Nel prossimo

esempio utilizzeremo questa libreria per dare un'ulteriore dimostrazione delle potenzialità dell'elemento `<canvas>`.

Close-pixelate carica l'immagine prescelta all'interno di un `<canvas>` e usa il Canvas API 2D per modificarne la visualizzazione.

### **LISTATO 6.23** Libreria close-pixelate: manipolazione di una foto attraverso Canvas API 2D

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Canvas: Manipolazione delle immagini</title>
    <style>
      body {margin: 2em; }
      canvas, img {border: 1px solid #666}
      figcaption { display: block; margin-bottom: 2em; font-family: arial, sans-serif; font-style: italic; }
    </style>
    <script src="close-pixelate.js"></script>
    <script src="image-forgery.js"></script>
    <script src="http://ajax.googleapis.com
    /ajax/libs/jquery/1.4.2/jquery.min.js"></script>
  </head>
  <body>
    <figure>
      
      <figcaption>Tre grazie minorchine</figcaption>
    </figure>
    <figure>
      
      <figcaption>Tre grazie minorchine</figcaption>
    </figure>
    <script>
      function canvasImg() {
        document.getElementById("minorchine").closePixelate([
          { resolution : 24 },
          { shape : "circle", resolution : 24, size: 16, offset: 12,
            alpha: 0.5 }
        ]);
      }
      if (document.addEventListener) {
        window.addEventListener("load", pixelize, false);
      }
    </script>
```

```
</body>  
</html>
```

Nel corpo del documento non troviamo che la stessa immagine riproposta due volte. La sola cosa che cambia è il valore assegnato all'attributo `id`. Questo ci consente di distinguere le immagini ai fini dell'identificazione del file che sarà incluso in un `<canvas>` e successivamente manipolato. Utilizziamo i nuovi elementi `<figure>` e `<figcaption>` secondo quanto illustrato nel Capitolo 2 per arricchire la semantica del documento.

Nell'intestazione della pagina web, ossia nel tag `<head>...</head>`, oltre all'abituale dichiarazione della codifica dei caratteri, alla definizione del titolo e a un set minimo di fogli di stile, si importano le tre librerie di cui abbiamo bisogno per poter applicare alla nostra foto le trasformazioni promesse da Close-pixelate.

In fondo al documento si associa l'invocazione della funzione `canvasImg()` all'evento `load`.

## addEventListener() in Internet Explorer 9

Internet Explorer, nelle versioni precedenti alla 9, ha adottato un modello di eventi diverso da quello standard. Questo si è tradotto, tra l'altro, nel mancato supporto del metodo `addEventListener()` in favore del proprietario `attachEvent()` che svolgeva una funzione identica. IE9, sin dalla versione beta, è il primo browser di casa Microsoft a supportare il metodo `addEventListener()`. Poiché la libreria Close-pixelate, al momento, non funziona con versioni del browser precedenti alla 9, e in virtù del fatto che tali versioni non riconoscono il metodo `addEventListener()`, se ne testa il supporto.

La funzione `canvasImg()` non fa altro che ottenere un riferimento all'immagine da elaborare per poi richiamare la funzione `closePixelate()` impostando alcuni parametri che produrranno l'effetto apprezzabile in Figura 6.12. Una spiegazione dettagliata del funzionamento della libreria Close-pixelate è al di fuori dallo scopo di questo manuale, tuttavia informazioni ulteriori sono disponibili presso: <http://desandro.com/resources/close-pixelate/>

### ATTENZIONE

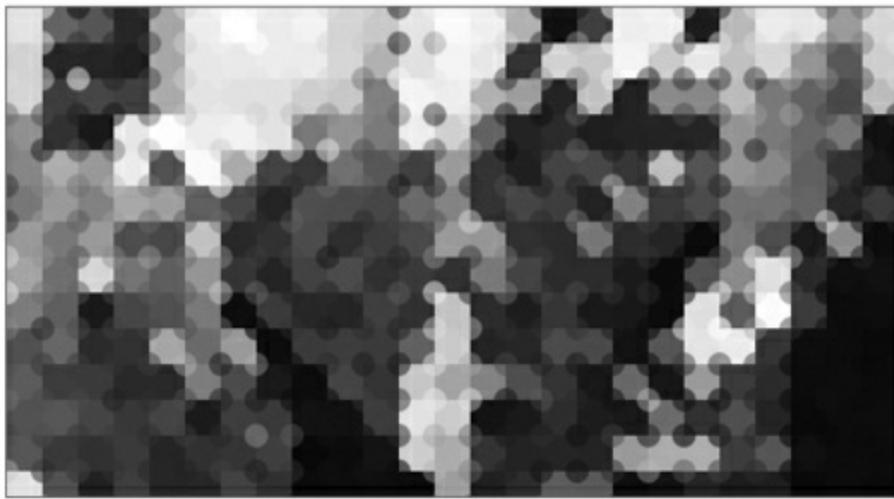
Qui vale la pena notare che questa tecnica di alterazioni delle immagini produce effetti artistici più efficaci quando il soggetto è inquadrato in primo piano.

## Testo

Il testo è un altro elemento che è possibile disporre all'interno del `<canvas>`. Lo script che segue mette in mostra i metodi e le proprietà a nostra disposizione per introdurre del testo all'interno di un oggetto canvas.



Tre grazie minorchine



Tre grazie minorchine

**FIGURA 6.12** La libreria Close-pixelate inserisce l'immagine in un <canvas> e ne modifica i pixel.

**LISTATO 6.24** Inserire un testo nel canvas

```
<script>
var contextSx = document.getElementById("sx").getContext("2d");
contextSx.font = "50px sans-serif";
contextSx.textAlign = "start";
contextSx.textBaseline = "middle";
contextSx.fillStyle = "rgb(140, 140, 140)";
contextSx.fillText("Italia", 50, 100);
var contextDx = document.getElementById("dx").getContext("2d");
contextDx.font = "italic 4em helvetica, sans-serif";
contextDx.textAlign = "start";
contextDx.textBaseline = "top";
contextDx.strokeText("Italia", 50, 100);
</script>
```

Il risultato di questo listato è apprezzabile in Figura 6.13.



Italia

Italia

**FIGURA 6.13** Testo nel canvas.

Volendo introdurre del testo in un oggetto canvas è possibile intervenire sui vari aspetti che lo riguardano, assegnando degli opportuni valori alla proprietà `font`. È bene qui ricordare come sia possibile impostare gli stessi valori impiegati per definire la proprietà `font` in CSS.

### CSS: la proprietà font

In CSS è possibile definire stile, dimensione, tipo di un carattere e altro ancora sia utilizzando le proprietà specifiche `font-style`, `font-size`, `font-weight`, `font-family`, `font-variant`, sia riconducendo tutti questi valori in capo a un'unica proprietà `font`.

Per l'allineamento del testo si impiega la proprietà `textAlign`, che ammette i seguenti valori: `"start"`, `"end"`, `"left"`, `"right"`, `"center"`. Qualora questa proprietà non sia esplicitamente impostata assumerà `"start"` come valore predefinito.

`textBaseline` è la proprietà che permette di intervenire sull'allineamento verticale del testo rispetto alle coordinate date nei metodi `fillText()` o `strokeText()`. I valori possibili sono: `"top"`, `"hanging"`, `"middle"`, `"alphabetic"`, `"ideographic"`, `"bottom"`. Il valore predefinito in mancanza di esplicita diversa assegnazione è `"alphabetic"`.

I metodi `fillText()` e `strokeText()` hanno lo scopo di definire il testo (primo argomento) da riportare nel canvas e le coordinate in cui esso apparirà (secondo e terzo argomento).

## Brevi considerazioni su SVG

Abbiamo visto come per mezzo del marcatore `<canvas>` e della relativa interfaccia di programmazione possiamo manipolare le immagini (la libreria Close-pixelate, a detta dello stesso autore, altro non è se non un modo di mettere in mostra le funzionalità dell'oggetto `ImageData`); la stessa interfaccia può essere utilizzata con profitto per riprodurre grafica vettoriale (le figure geometriche rappresentate nelle Figure 6.3, 6.5 e 6.7 ne costituiscono alcuni semplici esempi). In questo contesto tuttavia può essere utile valutare, in alternativa, SVG (acronimo di Scalable Vector Graphics, ossia grafica vettoriale scalabile). A titolo di esempio, riporto nel Listato 6.25 il codice SVG utile a

definire lo stesso effetto riprodotto con il gradiente di tipo radiante proposto nel Listato 6.16.

### **LISTATO 6.25** Figura geometrica con gradiente riprodotta in SVG

```
<svg>
<defs>
  <radialGradient id="gradient" cx="43%" cy="43%" r="45%" fx="35%" fy="35%">
    <stop offset="0%" style="stop-color:rgb(255,255,255); stop-opacity:0"/>
    <stop offset="50%" style="stop-color:rgb(204,204,204); stop-opacity:0.5"/>
    <stop offset="100%" style="stop-color:rgb(0,0,0); stop-opacity:1"/>
  </radialGradient>
</defs>
  <circle cx="50%" cy="50%" r="100" fill="url(#gradient)">
</svg>
```

### **NOTA**

All'interno della coppia di marcatori `<svg>...</svg>` si usa il linguaggio XML SVG che segue sue proprie regole dettate da una relativa specifica.

A prima vista sembra di poter affermare che esistano aree di sovrapposizione tra queste due tecnologie, in realtà si tratta di strumenti diversi che danno il loro meglio in contesti ben distinti. La Tabella 6.3 mira a sintetizzare il confronto.

### **TABELLA 6.3** Canvas e SVG a confronto

<b>CANVAS</b>	<b>SVG</b>
Buon supporto tra i principali browser presenti sul mercato.	Supporto non ottimale tra i browser.
Offre il suo meglio nella manipolazione dei pixel.	Offre il suo meglio nella rappresentazione di grafica vettoriale.
Non accessibile via DOM. Le figure disegnate per mezzo dell'interfaccia di programmazione Canvas 2D API non rientrano nella rappresentazione a oggetti del documento (DOM), per questo motivo non possono essere manipolate.	Accessibile via DOM. Le figure geometriche sono create attraverso un linguaggio dichiarativo i cui elementi sono parte integrante della struttura logica del documento HTML.
Una difficile gestione degli eventi ne scoraggia l'uso per quelle applicazioni che richiedono forme intensive di interazione .	Gestione più semplice degli eventi che ne suggerisce l'impiego per applicazioni interattive.
In termini di prestazioni è più efficiente di SVG.	Meno efficiente del canvas anche a causa della rappresentazione a oggetti delle figure geometriche descritte.

Difficile da utilizzare senza l'ausilio di libreria JavaScript di supporto.

È possibile disegnare le figure geometriche con un software di grafica vettoriale come Adobe Illustrator o Inkscape e poi esportare il proprio lavoro direttamente in SVG. Del resto questa è una scelta quasi obbligata perché non è semplice realizzare figure geometriche complesse senza ricorso a strumenti software.

## Riferimenti alle risorse citate e altri link utili

- La specifica W3C dell'elemento `<canvas>`: <http://www.w3.org/TR/html5/the-canvas-element.html>
- La specifica dell'interfaccia di programmazione del contesto bidimensionale del canvas Canvas 2D Context: <http://www.w3.org/TR/2010/WD-2dcontext-20100624>
- La suggestiva animazione di cui si dà conto in apertura è disponibile presso l'indirizzo: <http://labs.hyperandroid.com/js1k>
- "HTML5 Canvas per Internet Explorer". È con questo motto che si presenta la libreria JavaScript ideata da Google per sopperire al mancato supporto dell'elemento nelle versioni del browser precedenti alla 9.
- Per avere maggiori informazioni su Khronos Group e la specifica WebGL visitate: <http://www.khronos.org/webgl>
- La definizione di radiante pubblicata su Wikipedia: <http://it.wikipedia.org/wiki/Radiante>
- Close-pixelate è una libreria JavaScript scritta da David De Sandro che sfrutta l'interfaccia di programmazione applicata al marcatore `<canvas>` per produrre effetti artistici interessanti: <http://desandro.com/resources/close-pixelate>
- La specifica del linguaggio SVG: <http://www.w3.org/TR/SVG>
- L'elemento `<svg>` nella specifica HTML5: <http://www.w3.org/TR/html5/the-map-element.html#svg-0>
- Le performance di Canvas e SVG sono messe a confronto con una serie di test: <http://www.borismus.com/canvas-vs-svg-performance>
- Per realizzare grafiche vettoriali si suggerisce l'utilizzo di Inkscape, software disponibile sotto licenza GNU: <http://inkscape.org/download/>

## Conclusioni

In questo capitolo si è analizzata una parte importante della specifica Canvas API 2D. L'elemento `<canvas>` dispone già oggi di una discreta base di supporto presso i browser più popolari. Librerie come Excanvas ne estendono ulteriormente la diffusione, riuscendo a coprire in buona parte le sue funzionalità anche su browser non più recentissimi come Internet Explorer 6. Altre librerie si stanno diffondendo e contribuiscono a fare dell'oggetto canvas un elemento sempre più versatile, tanto per le applicazioni artistiche quanto per quelle professionali: applicazioni prima d'ora al di fuori dal dominio dell'HTML.

# Microdata: la rivoluzione silenziosa

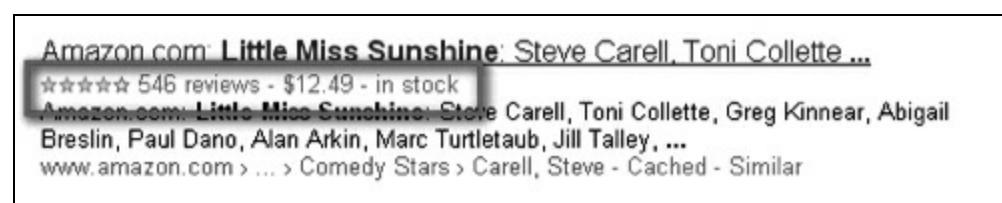
«Il Web è stato progettato come uno spazio di informazioni, tale da essere utile per le comunicazioni tra esseri umani, ma realizzato in modo che anche le macchine sarebbero state in grado di partecipare e aiutare.» Tim Berners-Lee

[Tratto da <http://www.w3.org/DesignIssues/Semantic.html>]

Come autori di pagine web pubblichiamo documenti ricchi di dati. Per la maggior parte dei casi si tratta di dati non strutturati che, finora, non siamo stati in grado di "vedere", "sentire", "catturare", perché persi in una massa informe. Ciò continua a essere vero anche a fronte di un uso accorto dei nuovi elementi semantici introdotti nel Capitolo 2 di questo manuale.

In assenza di forme di annotazione adeguate, estrarre e riutilizzare dati comporta grande dispendio di energie, il tutto per lo sviluppo di soluzioni ad hoc, poco riutilizzabili e con alti costi di manutenzione. La parte della specifica HTML5 che va sotto il nome di microdata rende possibile contrassegnare in maniera consistente informazioni come recensioni, eventi, dati personali e molto altro ancora, in modo da facilitarne la comprensione e conseguentemente l'impiego da parte di servizi automatici, quali i servizi di indicizzazione e aggregazione dati.

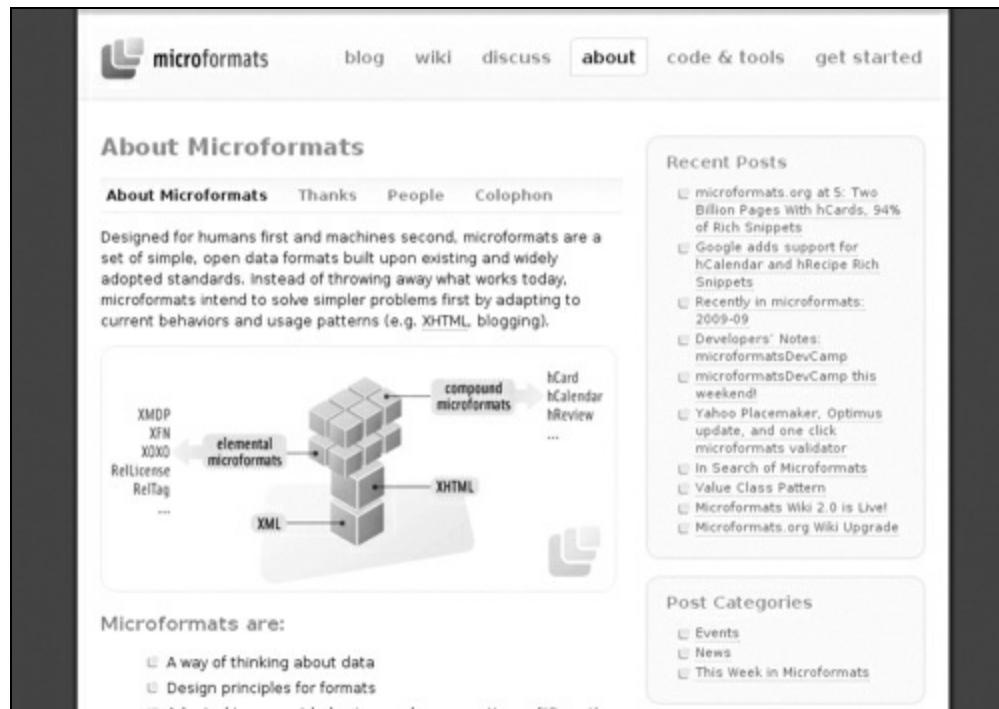
Google Rich Snippets (Figura 7.1) è un chiaro esempio di come i dati da cui è composta una pagina web possano tradursi in informazioni distribuibili secondo modalità diverse.



**FIGURA 7.1** Google Rich Snippets mostra un voto aggregato, il numero di recensioni che lo hanno prodotto e, nel caso di Amazon, anche prezzo e informazioni sulla disponibilità in magazzino del titolo cercato.

Perché farlo? Perché esporre a terzi i propri dati? Se vi ponete queste domande, forse, dovreste fare un passo indietro e domandarvi perché pubblicate sul Web! Semplificare il processo di raccolta ed elaborazione dei dati si rivela spesso una scelta vincente. Se invece vi chiedete: "perché adesso?", allora sappiate che Google Rich Snippets è già in grado di interpretare le informazioni addizionali rese disponibili tramite microdata. Con questa modalità di rappresentazione dei risultati di ricerca Google ha regalato un

incremento significativo di accessi ai collegamenti ipertestuali così proposti. Nel corso di questo capitolo vedremo come sfruttarne le potenzialità. Il tutto è possibile senza spendere un centesimo in più o meno efficaci consulenze degli strateghi del SEO (Search Engine Optimization): il risultato si ottiene solo contrassegnando i propri dati in modo più accurato. D'altro canto, considerare questo formato solo alla stregua di un mezzo per migliorare il posizionamento e la visibilità del proprio sito è riduttivo.



**FIGURA 7.2** La sezione “about” del sito relativo al formato Microformats.

Prima che questa parte della specifica vedesse la luce, altre tecnologie si sono affermate in quest’area; mi riferisco al formato Microformats (Figura 7.2) e all’RDFa, acronimo di Resource Description Framework in attributes (Figura 7.3): Microdata HTML5 può essere grossolanamente considerato come una versione ridotta e semplificata dell’RDFa; più precisamente, è uno strumento che contrassegna i dati per mezzo di un vocabolario realizzato su misura e referenziato nelle pagine web in cui i dati sono proposti secondo uno schema di coppia proprietà-valore.

are categorization labels. The gap between what programs and humans understand is large.



On the left, what browsers see. On the right, what humans see. Can we bridge the gap so browsers see more of what we see?

What if the browser received information on the meaning of a web page's visual elements? A

**FIGURA 7.3** Un estratto del testo introduttivo su RDFa pubblicato dal W3C. A sinistra ciò che vede il browser, a destra ciò che vede un umano. RDFa riduce questo gap.

## Microdata: la sintassi

La sintassi necessaria per marcare i dati pubblicati con il formato microdata è molto semplice. Con gli esempi di seguito illustrati procediamo, per approssimazioni successive, a trasformare una porzione di codice di puro HTML in un codice più composito, in cui le due tecnologie cooperino: del resto microdata è concepito per integrarsi con il linguaggio di marcatura ed esaltarne le proprietà semantiche.

### **LISTATO 7.1** Un ordinario paragrafo HTML

```
<p>
  Ciao, sono Mario, ma tutti mi chiamano MarioUnMartini, lavoro come cameriere presso il Bar Beque.
  Sono nato a Milano il <time datetime="1987-06-05">5 giugno 1987</time>. Nel
  tempo libero mi dedico al mio hobby preferito: la meccanica quantistica. Se vuoi
  saperne di più sul mio hobby visita il sito <a
  href="http://meccanicaquantisticafrottola.org/" title="La Meccanica
  Quantistica che mia nonna capisce">Meccanica Quantistica Mica Frottola</a>
</p>
```

L'esuberante figura di Mario, barista appassionato di meccanica quantistica, è il soggetto di questo paragrafo. Dal punto di vista del formato microdata, possiamo pensare a Mario come all'elemento (in inglese item) con riferimento al quale identificheremo una serie di proprietà e di valori corrispondenti.

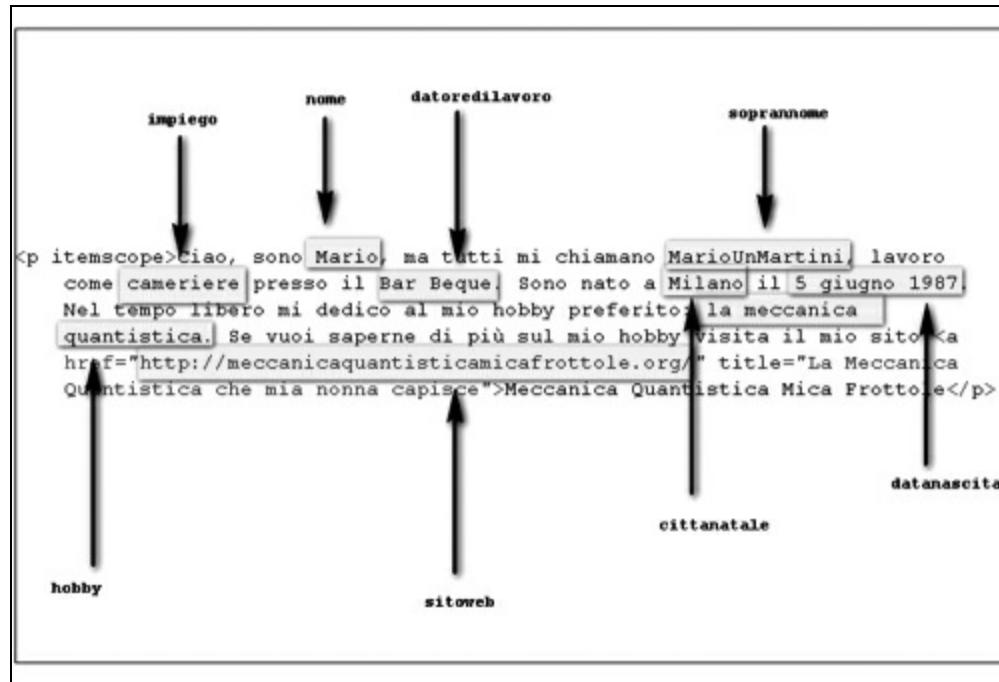
Il primo passo sarà dunque contrassegnare questo paragrafo come un elemento.

### **LISTATO 7.2** Una porzione di codice HTML

```
<p itemscope>Ciao, sono Mario... </p>
```

itemscope è un attributo globale: ciò significa che possiamo applicarlo a un qualunque marcatore HTML. Esso contrassegna un insieme di proprietà come logicamente legate tra loro al fine di rappresentare un dato elemento. Con questa annotazione diciamo: all'interno di questo paragrafo sono presenti informazioni relative a un non meglio precisato elemento.

Mario merita certo un profilo più definito di quello attuale. Lo step successivo consisterà nell'identificare le proprietà di questo elemento: passiamo in rassegna questo paragrafo.



**FIGURA 7.4** I valori delle proprietà sono evidenziati nel testo.

Per contrassegnare in modo opportuno le proprietà che abbiamo individuato faremo ricorso al tag `<span>..</span>`, al quale applicheremo l'attributo globale `itemprop` che è utile per identificare una proprietà.

Prendiamo per esempio la prima proprietà: il nome.

#### **LISTATO 7.3** Definire una proprietà microdata

```
<p><span itemprop="nome">Mario</span>...[omissis] </p>
```

Così facendo abbiamo appena definito la proprietà nome dell'elemento e ne abbiamo impostato il suo valore a "Mario". Generalizzando potremmo scrivere:

#### **LISTATO 7.4** Regola generale di definizione di una proprietà microdata

```
<span itemprop="nome_della_proprietà">valore_della_proprietà</span>
```

dove `nome_della_proprietà` è un segnaposto che andrà sostituito con il nome effettivo della proprietà che si vuole definire. `valore_della_proprietà` è un altro segnaposto che andrà sostituito con il suo effettivo valore.

Possiamo applicare questa regola generale a molte delle altre proprietà evidenziate in Figura 7.4.

#### **LISTATO 7.5** Alcune proprietà del nostro elemento

```
<span itemprop="nome">Mario</span>
<span itemprop="soprannome">MarioUnMartini</span>
<span itemprop="impiego">cameriere</span>
<span itemprop="datoredilavoro">Bar BeQue</span>
<span itemprop="cittanatale">Milano</span>
<span itemprop="hobby">meccanica quantistica</span>
```

Esistono eccezioni alla regola sopra enunciata. In alcuni casi il valore della proprietà non è da ricercarsi nel testo racchiuso tra i tag HTML bensì tra uno dei suoi attributi. Abbiamo due esempi in questo paragrafo. Prendiamo il caso della data di nascita di Mario. La data appare due volte: sembrerebbe dunque legittimo prendere in considerazione, come per tutti gli altri casi, il testo contenuto all'interno del tag. Tuttavia bisogna ricordare che queste proprietà contrassegnano dati che idealmente saranno manipolati da macchine e non da persone. Per questo motivo la rappresentazione della data espressa dal valore assegnato all'attributo `datetime` è più efficace di "5 giugno 1987".

#### **LISTATO 7.6** Il valore della proprietà datanascita

```
<time itemprop="datanascita" datetime="1987-06-05">5 giugno 1987</time>
```

Troviamo un altro esempio nello stesso paragrafo in occasione del collegamento ipertestuale al sito web di Mario:

#### **LISTATO 7.7** Il valore della proprietà sitoweb

```
<a href="http://meccanicaquantisticafrottole.org/" title="La Meccanica Quantistica che mia nonna capisce">Meccanica Quantistica Mica Frottole</a>
```

Anche in questo caso, ciò che conta è l'indirizzo del sito web. Si potrebbe reperire la stessa informazione anche in formato testo al posto del più accattivante titolo "Meccanica Quantistica Mica Frottole", ma se si vuole avere la ragionevole certezza di estrarre l'URL del sito è nel valore dell'attributo `href` che bisogna cercare.

## Altre eccezioni

Oltre ai tag `<time>` e `<a>` qui citati, esistono altri casi in cui il valore della proprietà microdata che desideriamo impostare non è da ricercare nel testo racchiuso tra il tag di apertura e quello di chiusura. Un altro esempio è quello di un'immagine. In questo caso, il valore della proprietà microdata è definito, come presumibilmente avrete già compreso, mediante l'attributo `src`. Sebbene possa sembrare complesso, in realtà è solo un esercizio di buon senso. Non vi sarà difficile identificare caso per caso dove cercare per trovare il valore oggetto dell'annotazione.

Ricapitolando, è possibile introdurre le annotazioni alla base del formato microdata operando come descritto nel prossimo listato.

### **LISTATO 7.8** Paragrafo integrato con formato microdata

```
<p itemscope>
  Ciao, sono
  <span itemprop="nome">Mario</span>,
  ma tutti mi chiamano
  <span itemprop="soprannome">MarioUnMartini</span>,
  lavoro come
  <span itemprop="impiego">cameriere</span>
  presso il
  <span itemprop="datorelavoro">Bar BeQue</span>.
  Sono nato a
  <span itemprop="cittanatale">Milano</span>
  il <time itemprop="datanascita" datetime="1987-06-05">5 giugno
  1987</time>.
  Nel tempo libero mi dedico al mio hobby preferito: la
  <span itemprop="hobby">meccanica quantistica</span>.
  Se vuoi saperne di più sul mio hobby visita il sito
  <a itemprop="http://meccanicaquantisticafrottole.org/" href="http://meccanicaquantisticafrottole.org/" title="La Meccanica Quantistica che mia nonna capisce">Meccanica
  Quantistica Mica Frottole</a>
</p>
```

## Alcuni casi speciali

Fin qui abbiamo appreso le basi dell'annotazione secondo il formato microdata. Esistono tuttavia alcuni casi, appena più complessi di quelli esaminati nei paragrafi precedenti, che potrebbero presentarsi dinanzi a noi nel momento in cui avremo intenzione di applicare questo formato di annotazione dei dati a scenari diversi. Come comportarsi quando proprietà diverse ammettono lo stesso valore? Che cosa succede nel caso contrario in cui una stessa proprietà debba vedersi attribuito più di un valore? Si possono innestare più elementi? E ancora, che cosa succede se la proprietà di un elemento non si trova gerarchicamente al di sotto della definizione di elemento microdata?

## Caso 1: proprietà diverse hanno uno stesso valore

Può accadere che proprietà diverse tra loro accettino uno stesso valore. Ipotizziamo che Mario dia una svolta alla sua carriera e riesca a trasformare il suo hobby anche in un lavoro:

### **LISTATO 7.9** Stessa proprietà, valori diversi: un caso concreto

Da qualche tempo ho cambiato mestiere, ora il mio hobby è diventato anche materia per il mio lavoro: scriverò un saggio sulla

```
<span itemprop="hobby lavoro">meccanica quantistica</span>.
```

La specifica sul formato microdata propone un altro esempio meno surreale di questo, che può essere di ulteriore aiuto alla comprensione:

### **LISTATO 7.10** Stessa proprietà, valori diversi: un caso concreto

```
<div itemscope>
<span itemprop="favorite-color favorite-fruit">orange</span>
</div>
```

In entrambi i casi è comunque evidente come le due proprietà appaiano, separate da uno spazio, come valore dell'attributo `itemprop`.

## Caso 2: stessa proprietà, valori diversi

È possibile ripetere una stessa proprietà tante volte quanti sono i valori che si vuole che assuma. Ipotizzando che Mario voglia dividere il suo tempo libero tra la meccanica quantistica e il parapendio, avremmo potuto esprimere questa informazione come segue.

### **LISTATO 7.11** Stessa proprietà, valori diversi: un caso concreto

Divido il mio tempo libero tra i due miei hobbies preferiti: la  
<span itemprop="hobby">meccanica quantistica</span>  
e il <span itemprop="hobby">parapendio</span>.

## Caso 3: elementi innestati

Il formato microdata prevede che la dichiarazione di un elemento possa convivere all'interno di un altro elemento.

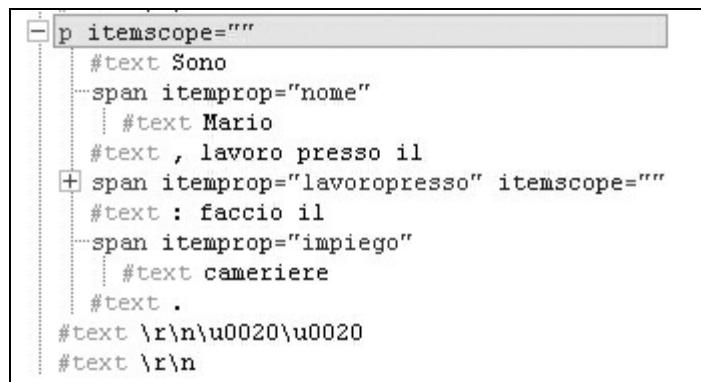
### **LISTATO 7.12** Elementi innestati: un esempio

```

<p itemscope>
  Sono
  <span itemprop="nome">Mario</span>
  , lavoro presso il
  <span itemprop="lavoropresso" itemscope>
    <span itemprop="denom-sociale">Bar BeQue</span>,
    il mio datore di lavoro si chiama
    <span itemprop="datore-lavoro">Gigi</span>
  </span>
  : faccio il
  <span itemprop="impiego">cameriere</span>
</p>

```

La rappresentazione del modello del documento a oggetti (DOM) ci aiuta nell'analisi di questo caso (Figura 7.5).



**FIGURA 7.5** Rappresentazione dei nodi DOM ottenuti a partire dal paragrafo.

Il paragrafo, per mezzo dell'attributo `itemscope`, definisce un elemento: si tratta di Mario. A lui fanno capo le tre proprietà definite, ossia `nome`, `lavoropresso` e `impiego`. La proprietà `lavoropresso` è in realtà essa stessa un elemento, come l'attributo `itemscope` lascia intuire. Infatti, esplodendo il simbolo posto alla sinistra del nodo rappresentato dal marcatore `<span>` troviamo la rappresentazione sintetizzata in Figura 7.6.

```

p itemscope=""
  #text Sono
  span itemprop="nome"
    #text Mario
  #text , lavoro presso il
  span itemprop="lavoropresso" itemscope=""
    span itemprop="denom-sociale"
      #text Bar BeQue
    #text , il mio datore di lavoro si chiama
    span itemprop="datore-lavoro"
      #text Gigi
    #text : faccio il
    span itemprop="impiego"
      #text cameriere
    #text .
#text \r\n\u0020\u0020
#text \r\n

```

**FIGURA 7.6** Elementi microdata innestati: una vista dei nodi del DOM.

Da ciò risulta evidente come esista un elemento interno dotato anch'esso di sue proprietà specifiche: `denom-sociale` e `datore-lavoro`.

#### Caso 4: proprietà dichiarate al di fuori dall'elemento

Se guardiamo al DOM prodotto da questa porzione di codice (Figura 7.7), scopriamo che tutte le proprietà dell'elemento che abbiamo così definito sono innestate al suo interno: si trovano tutte all'interno del paragrafo.

```

#document
  HTML
  htm
    head
    body
      #text \n\u0020\u0020\n
      p itemscope=""
        #text Ciao, sono
        span itemprop="nome"
        #text , ma tutti mi chiamano
        span itemprop="soprannome"
        #text , lavoro come
        span itemprop="impiego"
        #text presso il
        span itemprop="datorelavoro"
        #text . Sono nato a
        span itemprop="cittanatale"
        #text il
        time itemprop="datanascita" datetime="1987-06-05"
        #text . Nei tempo libero mi dedico al mio hobby preferito: la
        span itemprop="hobby"
        #text . Se vuoi saperne di più sul mio hobby visita il sito
        a itemprop="" href="http://meccanicequantisticamericafrottole.org/" title="La Meccanica
        Quantistica che mia nonna capisce"
        #text \n\n\u0020\u0020
        #text \n

```

**FIGURA 7.7** Tutti i nodi relativi alle proprietà dell'elemento si trovano all'interno del paragrafo che lo definisce.

In un caso come questo tutte le proprietà dell'elemento sono implicitamente riconducibili allo stesso, perché contenute al suo interno. È, invece, necessario esplicitare questa relazione di appartenenza quando le proprietà sono dichiarate al di fuori dall'elemento che contribuiscono a definire. A questo scopo si usa l'attributo `itemref`.

```
<p itemscope>
  Sono
  <span itemprop="nome">Mario</span>
  <span itemprop="lavoropresso" itemscope itemref="lavoropresso-
  rif"></span>
  , faccio il <span itemprop="impiego">cameriere</span>
</p>
<p id="lavoropresso-rif">
  Lavoro presso il
  <span itemprop="denom-sociale">Bar BeQue</span>
  , il mio datore di lavoro si chiama
  <span itemprop="datore-lavoro">Gigi</span>
</p>
```

Il primo paragrafo definisce un elemento microdata. Una delle proprietà di questo elemento si chiama `lavoropresso`. Questa proprietà è, a sua volta, un elemento microdata. Le proprietà di quest'ultimo elemento non sono definite all'interno del primo paragrafo ma in un altro a sé stante. Per mantenere una relazione tra la definizione dell'elemento microdata che si trova nel primo paragrafo e le sue proprietà definite invece nel secondo usiamo l'attributo `itemref`. Il suo valore rappresenterà il valore assegnato all'attributo `id` del marcatore HTML in cui le proprietà dell'elemento sono definite. `itemref` così rappresenta il collegamento tra la definizione dell'elemento microdata e la definizione delle sue proprietà (Figura 7.8).



```
<p itemscope>Sono <span itemprop="nome">Mario</span> <span itemprop="lavoropresso" itemscope itemref="lavoropresso-rif"></span>, faccio il <span itemprop="impiego">cameriere</span>.</p>
<p id="lavoropresso-rif">Lavoro presso il <span itemprop="denom-sociale">Bar BeQue</span>, il mio datore di lavoro si chiama <span itemprop="datore-lavoro">Gigi</span></p>
```

**FIGURA 7.8** itemref collega l'elemento con le sue proprietà.

## Elementi tipizzati

Nei paragrafi precedenti abbiamo analizzato la sintassi del formato microdata. Tuttavia, limitandoci ad applicare le regole sin qui apprese possiamo tutt'al più produrre un codice microdata che presenta forti limitazioni nel suo riutilizzo. Esiste la concreta possibilità di incorrere in un conflitto con elementi microdata definiti da altri autori. Vi è un'ambiguità che dobbiamo risolvere se vogliamo favorire il riutilizzo delle informazioni così contrassegnate. La specifica prevede che questo problema possa essere risolto ricorrendo alla "tipizzazione" degli elementi microdata; recita testualmente: "Il tipo delinea il

contesto delle proprietà, definendo così un vocabolario: una proprietà chiamata 'classe' attribuita a un elemento il cui tipo sia <http://census.example/person> potrebbe riferirsi alla classe economica di un individuo, mentre la proprietà 'classe' applicata a un elemento di tipo <http://example.com/school/teacher> potrebbe riferirsi alla classe in cui il professore imparte le sue lezioni".

I termini del problema così come presentati dalla specifica sono chiari, ma perché si fa riferimento a quegli indirizzi web? Partiamo dal concetto, che a questo punto dovrebbe essere esplicito: se privo di un "tipo", l'elemento microdata non è di fatto riutilizzabile. Applicare un "tipo" è semplice: basta esplicitare un valore per l'attributo `itemtype`.

#### **LISTATO 7.14** `itemtype`: l'importanza di un identificativo univoco

```
<p itemscope itemtype="identificativo_univoco">  
  Ciao, sono  
  <span itemprop="nome">Mario</span> ... [omissis]  
</p>
```

Se "tipizzare" un elemento è un'attività che ha senso al solo fine di risolvere una potenziale ambiguità tra nomi di proprietà identici ma che si riferiscono a elementi diversi, allora diventa di cruciale importanza scegliere, per demarcare il tipo, un identificatore che sia univoco. Un modo semplice e abbastanza diffuso prevede di scegliere un URL sotto il proprio controllo. Se sei il proprietario del dominio [example.org](http://example.org) puoi scegliere come identificativo per un elemento che rappresenti una persona il seguente URL: <http://www.example.org/Persona>.

## Domini come identificatori univoci in programmazione

Se avete confidenza con il linguaggio di programmazione Java, questa non sarà per voi una scoperta epocale, anzi non sarà affatto una scoperta. In Java, ma la stessa prassi è in uso anche presso .NET, il package all'interno del quale viene definita ogni classe è composto dal dominio della propria azienda a elementi invertiti. Per esempio:

```
package org.example; class TestMePlease {    public static void main(String args[]) {  
    System.out.println("Hello World!");    } code}
```

## Supporto

Quella parte della specifica relativa al formato microdata prevede, oltre agli attributi sin qui analizzati, anche un'interfaccia di programmazione utile per la manipolazione dei dati contrassegnati nel rispetto di tale formato attraverso il linguaggio di scripting. Sfortunatamente tale interfaccia incontra ancora un supporto inesistente presso i principali browser in commercio. Per testarne il supporto non possiamo fare ricorso alla libreria Modernizr utilizzata allo scopo in relazione ad altre funzionalità. Non ci resta che mettere alla prova le versioni future dei browser con questo semplice script, che si basa su di un test di esistenza del metodo `getItems()`.

```
if (!document.getItems) {  
  // microdata API supportata
```

```
} else {  
    // supporto inesistente per microdata API  
}
```

Il metodo `getItems()`, se supportato, restituisce una lista di nodi relativi agli elementi microdata presenti nel documento HTML.

Poiché il tipo di un elemento ne definisce il vocabolario, appare evidente l'importanza di utilizzare vocabolari che siano quanto più diffusi e condivisi. Ciò si rende necessario al fine di favorire una reale condivisione delle informazioni. Anche in quest'ottica risulta utile prendere familiarità con Google Rich Snippets e con il vocabolario che questo progetto sta costruendo.

## Google Rich Snippet: le informazioni al servizio dei risultati di ricerca

Google Rich Snippets è una modalità di rappresentazione dei risultati del famoso motore di ricerca che fa leva sull'utilizzo dei dati strutturati messi a disposizione dagli autori di pagine web. Il vantaggio per chi effettua la ricerca è quello di ottenere un set più preciso di informazioni (al posto dell'ordinario estratto del testo della pagina), con il quale si può fare un'idea più precisa di quello che troverà una volta attivato il collegamento ipertestuale.

Google ha rilevato come i link che offrono una sintesi ben organizzata delle proprie informazioni abbiano sperimentato un incremento di traffico significativo. Possiamo dedurne che lo sforzo richiesto dall'annotazione addizionale dei dati si traduce in concreto in un beneficio non solo per gli utenti di Google ma anche per la popolarità del proprio sito web.

Google Rich Snippets nasce per interpretare formati come RDFa e Microformats. Solo più di recente, a partire dal 12 maggio 2009, ha esteso la sua capacità di interpretazione anche alla specifica del formato microdata. Sostanzialmente sono coperti i formati di codifica delle informazioni oggi più diffusi sul Web. Più di ogni altro aspetto, Google sta curando anche la creazione di un vocabolario che cresce gradualmente per numero di proprietà ed elementi definiti.

Rivediamo le nozioni apprese in questo capitolo per capire come poterle utilizzare in modo che Google comprenda meglio le informazioni che pubblichiamo. Google Rich Snippets, a oggi, è in grado di riconoscere e interpretare solo le informazioni che si riferiscono a:

- persone;
- recensioni singole o aggregate;
- eventi;

- aziende;
- ricette;
- prodotti.

Analizziamo nel dettaglio i primi due tipi di dati: persone e recensioni. Una volta compreso come procedere, non sarà difficile applicare un codice di annotazione che funzioni anche per le altre casistiche.

## Persone

In Figura 7.9 è illustrato un esempio di come Google organizza i dati che interpreta attraverso le annotazioni microdata (ma RDFa e Microformats sono formati alternativi per ottenere lo stesso risultato).

**Reid Hoffman - LinkedIn**    
 San Francisco Bay Area - Entrepreneur. Product Strategist. Investor.  
 View **Reid Hoffman's** professional profile on LinkedIn. LinkedIn is the world's largest business network, helping professionals like **Reid Hoffman** discover ...  
[www.linkedin.com/in/reidhoffman](http://www.linkedin.com/in/reidhoffman) - Cached

**FIGURA 7.9** LinkedIn annota i propri dati, permettendo una sintetica ed efficace rappresentazione degli stessi.

La prima riga, subito sotto il link, presenta una serie di informazioni relative al profilo LinkedIn di Reid Hoffman, co-fondatore del social network per professionisti.

Per capire come annotare tali informazioni in modo da enfatizzarle, possiamo prendere visione del vocabolario che Google ha realizzato per contrassegnare dati relativi a persone. Se puntiamo il browser all'indirizzo: <http://www.data-vocabulary.org/Person>, troviamo la pagina rappresentata in Figura 7.10.

Person	
• Looking for Google's <a href="#">webmaster docs</a> ?	
• Looking for <a href="#">B2C</a> ?	
See our <a href="#">Person webmaster docs</a> for full explanation of the Person type, and how to encode Person data in RDFa, microformats or microdata. This page exists to contain a machine-readable microdata description of the Person type itself (at the URI <a href="http://data-vocabulary.org/Person">http://data-vocabulary.org/Person</a> ).	
An item with the <code>item:type</code> <a href="http://data-vocabulary.org/Person">http://data-vocabulary.org/Person</a> represents a person. The following are the type's <a href="#">defined property names</a> .	
Property	Description
<code>name</code> (El)	Name
<code>nickname</code>	Nickname
<code>photo</code>	An image link
<code>title</code>	The person's title (for example, Financial Manager)
<code>role</code>	The person's role (for example, Accountant)
<code>url</code>	Link to a web page, such as the person's home page
<code>affiliation</code> (org)	The name of an organization with which the person is associated (for example, an employer). If <code>fn</code> and <code>org</code> have the exact same value, Google will interpret the information as referring to a business or organization, not a person.
<code>friend</code>	Identifies a social relationship between the person described and another person.
<code>contact</code>	Identifies a social relationship between the person described and another person.
<code>acquaintance</code>	Identifies a social relationship between the person described and another person.
<code>address</code> (adr)	The location of the person. Can have the subproperties <code>street-address</code> , <code>locality</code> , <code>region</code> , <code>postal-code</code> , and <code>country-name</code> .

**FIGURA 7.10** Le proprietà dell'elemento persona.

Si tratta di un elenco di proprietà che Google ha selezionato per definire una persona. Possiamo riscrivere il paragrafo con alcune note biografiche di Mario per generare un codice che sia poi interpretabile da Google. Dovremmo rinunciare alla notazione più ricca che ci siamo inventati. Il set di informazioni che riusciamo a rendere disponibili a terzi, in forma strutturata, utilizzando il vocabolario che Google ci mette a disposizione, è più ridotto, ma ampiamente riutilizzabile e non solo da Google.

## Non solo Google Rich Snippets!

È necessario tenere a mente che le informazioni annotate nel rispetto di questo vocabolario saranno in realtà messe a disposizione di chiunque voglia decidere di interpretarle. È plausibile aspettarsi che nuovi servizi online ed estensioni sviluppate per i principali browser traggano vantaggio da un insieme di dati più comprensibili perché annotati secondo un vocabolario condiviso e dunque dotati di un contesto noto.

### **LISTATO 7.15** Codice annotato secondo il vocabolario proposto da Google per le persone

```
<p itemscope itemtype="http://data-vocabulary.org/Person">  
  Ciao, sono  
  <span itemprop="name">Mario</span>  
  , ma tutti mi chiamano  
  <span itemprop="nickname">MarioUnMartini</span>,  
  lavoro come  
  <span itemprop="role">cameriere</span>  
  presso il  
  <span itemprop="affiliation">Bar BeQue</span>.  
  Nel tempo libero mi dedico al mio hobby preferito: la meccanica quantistica.  
  Se vuoi saperne di più sul mio hobby visita il sito  
  <a itemprop="url"  
    href="http://meccanicquantisticamicafrottole.org/"  
    title="La Meccanica Quantistica che mia nonna capisce">Meccanica  
    Quantistica Mica Frottole  
  </a>  
</p>
```

Google impiega solo una parte di queste informazioni per la visualizzazione nei risultati di ricerca. Per farci un'idea approssimativa di che cosa verrebbe mostrato nella pagina dei risultati possiamo usare uno strumento che Google mette a disposizione all'indirizzo

<http://www.google.com/webmasters/tools/richsnippets>.

#### **Rich Snippets Testing Tool** Beta

Use the Rich Snippets Testing Tool to check that Google can correctly parse your structured data markup and display it in search results.

##### **Test your website**

Enter a web page URL to see how it may appear in search results:

Examples: [UrbanSpoon](#), [LinkedIn](#), [Eventful](#), [Food Network](#)

## FIGURA 7.11 Lo strumento cui sottoporre il nostro codice.

È sufficiente indicare l'indirizzo della pagina web e fare clic sul pulsante Preview per ottenere un'anteprima: quella relativa al codice appena proposto verrebbe interpretata come illustrato in Figura 7.12.

Mario un barista prestato alla meccanica quantistica  
cameriere - Bar BeQue  
The excerpt from the page will show up here. The reason we can't show text from your  
webpage is because the text depends on the query the user types.  
html5.gigliotti.it/test/grs\_person.html - [Cached](#) - [Similar](#)

## FIGURA 7.12 Il nostro codice microdata come interpretato da Google.

La prima riga è estratta dal titolo del documento: fin qui nulla di nuovo. È la seconda riga che deve destare la nostra attenzione: il ruolo di Mario e il bar in cui lavora sono enfatizzati solo grazie agli attributi aggiuntivi utilizzati per annotare con il formato microdata le informazioni pubblicate. La terza e la quarta riga, come riportato anche nel testo, possono cambiare a seconda delle parole chiave specifiche utilizzate da chi sta eseguendo la ricerca. L'ultima riga è, anche qui nulla di nuovo, l'URL della pagina web.

## Recensioni

Le recensioni, ossia i commenti che a vario titolo sono proposti in merito a libri, film, musica, ristoranti, e qualunque altra cosa su cui si possa esercitare il proprio libero pensiero, sono attualmente uno dei pochi elementi che Google ha deciso di inserire nel suo vocabolario. Le recensioni vengono proposte in due modalità: singole e aggregate.

### Recensioni singole

Di seguito propongo la recensione del bar in cui lavora Mario:

“Bar BeQue. Recensito da Gabriele Gigliotti il 16 luglio. Caffè strepitoso, servizio ottimo! La specialità del Bar è il caffè 100% di qualità Arabica offerto in esclusive tazzine dello stilista Marani. Particolarità: tra un caffè e l’altro sarete intrattenuti da interessanti dissertazioni sul principio di indeterminazione di Heisenberg. Voto: 5”

Per le recensioni singole facciamo riferimento al vocabolario disponibile all’indirizzo:

<http://www.data-vocabulary.org/Review>.

Applicando questo vocabolario alla recensione del Bar BeQue otteniamo:

**LISTATO 7.16** Codice annotato secondo il vocabolario proposto da Google per le recensioni singole

```

<div itemscope itemtype="http://data-vocabulary.org/Review">
  <span itemprop="itemreviewed">Bar BeQue</span>
  Recensito da <span itemprop="reviewer">Gabriele Gigliotti</span>
  il
  <time itemprop="dtreviewed" datetime="2010-07-16">16 luglio
  2010</time>.
  <span itemprop="summary">Caffè strepitoso, servizio ottimo!</span>
  <span itemprop="description">
    La specialità del Bar è il caffè 100% di qualità Arabica offerto in esclusive tazzine dello stilista Marani.
    Particolarità: tra un caffè e l'altro sarete intrattenuti da interessanti dissertazioni sul principio di
    indeterminazione di Heisenberg
  </span>
  Voto: <span itemprop="rating">5</span>
</div>

```

Property	Description
itemreviewed (item)	The item being reviewed. In microformats, can include the name of the item reviewed (e.g.).
rating	A numerical quality rating for the item (for example, 4). You can indicate a rating scale by specifying best: (default: 5) and worst: (default: 1). <a href="#">More information about review ratings.</a>
reviewer	The author of the review.
dtreviewed	The date that the item was reviewed in ISO date format.
description	The body of the review.
summary	A short summary of the review.

**FIGURA 7.13** Le proprietà della recensione singola.

Vediamo nel dettaglio come abbiamo impiegato le proprietà a nostra disposizione.

- **itemreviewed**: l'oggetto della nostra recensione, nel nostro caso si tratta del Bar BeQue.
- **reviewer**: chi effettua la recensione; qui ho inserito il mio nome e cognome.
- **dtreviewed**. Data e ora in cui la recensione è stata scritta. Per questo tipo di dato si usi sempre `<time>`: ha il vantaggio di poter esprimere lo stesso dato sia per la macchina (quanto proposto come valore per l'attributo `datetime`) sia per l'umano, ossia il testo annidato nel tag `<time>...</time>`.
- **summary**: una manciata di parole con cui sintetizzare la recensione.
- **description**: la recensione vera e propria.
- **rating**: il voto assegnato. Una sintesi numerica della recensione che va da un minimo di 1 (il voto peggiore) a 5. È prevista la possibilità di adottare scale diverse, per esempio, da 1 a 10, ma in tal caso dovreste preoccuparvi di comunicare non solo il voto ma anche la scala cui ricondurre questo voto.

In Figura 7.14 è illustrata un'anteprima della recensione ottenuta usando "Rich Snippets Testing Tool".

Recensione: Bar BeQue  
★★★★★ Review by Gabriele Gigliotti - Jul 16, 2010  
The excerpt from the page will show up here. The reason we can't show text from your webpage is because the text depends on the query the user types.  
html5.gigliotti.it/test/grs\_single\_review.html - [Cached](#) - [Similar](#)

**FIGURA 7.14** Un'anteprima del possibile risultato ottenuto a fronte dell'interpretazione del codice di marcatura riportato nel Listato 7.18.

## Recensioni aggregate

Le recensioni aggregate hanno senso di esistere per siti come Trip Advisor, Amazon e, più in generale, tutti quei siti che sono in grado di catalizzare un gran numero di recensioni riproponibili in forma sintetica attraverso un voto medio espresso nella scala da 1 a 5.

Il vocabolario proposto da Google per questo tipo di recensioni è visualizzabile all'indirizzo: <http://www.data-vocabulary.org/Review-aggregate/> (Figura 7.15). Come per gli altri elementi, la pagina mostra un elenco delle proprietà da utilizzare per sintetizzare questo aggregato di dati.

Review-aggregate

See our [Review-aggregate webmaster docs](#) for full explanation of the Review-aggregate type, and how to encode Review-aggregate data in RDFa, microformats or microdata. This page exists to contain a machine-readable microdata description of the Review-aggregate type itself (at the URI <http://data-vocabulary.org/Review-aggregate>). An item with the `item type` <http://data-vocabulary.org/Review-aggregate> represents a review of a business, product or organization. The following are the type's defined property names:

Property	Description
<code>itemreviewed</code> ( <code>item</code> )	The item being reviewed. In microformats, can contain the name of the item reviewed ( <code>ta</code> ).
<code>rating</code>	A numerical quality rating for the item (for example, 4). You can indicate a rating scale by specifying <code>best</code> (default: 5) and <code>worst</code> (default: 1). <a href="#">More information about review ratings.</a>
<code>count</code>	The total number of reviews for the item.
<code>votes</code>	Specifies the number of people who provided a rating with or without an accompanying review. A site can specify <code>count</code> or <code>vote</code> , or both.
<code>summary</code>	A short summary of the collection of reviews that are being aggregated.

**FIGURA 7.15** Le proprietà della recensione aggregata.

Ipotizziamo che siano state scritte 20 recensioni mentre per 42 volte è stato attribuito un voto; inoltre ipotizziamo che il voto medio di queste recensioni sia pari a 3,5. Sfruttando le proprietà pubblicate da Google possiamo scrivere un codice di questo tipo:

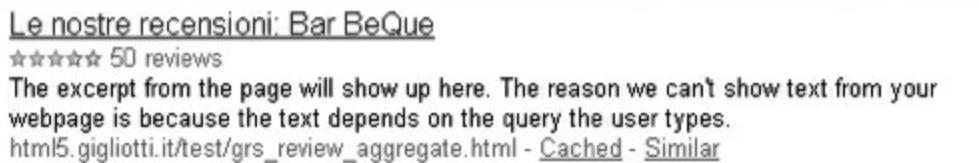
### LISTATO 7.17 Codice annotato secondo il vocabolario proposto da Google per le recensioni singole

```
<div itemscope itemtype="http://data-vocabulary.org/Review-aggregate">
  <span itemprop="itemreviewed">Bar BeQue</span>
  <span itemprop="rating">3.5</span>
```

```
voto medio espresso sulla base di <span itemprop="votes">42</span>  
voti.  
<span itemprop="count">20</span> recensioni.  
</div>
```

1. `itemreviewed`: ha lo stesso significato visto sopra per le recensioni singole.
2. `rating`: il significato è lo stesso che per la singola recensione, tuttavia in questo caso il numero espresso è una media dei singoli voti attribuiti all'oggetto di questa recensione.
3. `count`: indica il numero totale di recensioni disponibili su questo "oggetto".
4. `votes`: il numero totale di voti la cui media ha prodotto il rating.
5. `summary`: riporta un breve elenco delle sintesi di singole recensioni.

Ed ecco in Figura 7.16 il risultato prodotto da questo codice:



Le nostre recensioni: Bar BeQue  
★★★★★ 50 reviews  
The excerpt from the page will show up here. The reason we can't show text from your webpage is because the text depends on the query the user types.  
html5.gigliotti.it/test/grs\_review\_aggregate.html - [Cached](#) - [Similar](#)

**FIGURA 7.16** Anteprima di una recensione aggregata riprodotta a partire dal codice del Listato 7.17.

## Riferimenti alle risorse citate e altri link utili

- La specifica W3C relativa agli attributi che permettono di annotare i dati con microdata: <http://www.w3.org/TR/html5/microdata.html>
- Il sito ufficiale del formato Microformats: <http://microformats.org/>
- Testo introduttivo su RDFa pubblicato dal W3C: <http://www.w3.org/TR/xhtml-rdfa-primer/>
- Google Rich Snippet: una descrizione in lingua italiana: <http://www.google.com/support/webmasters/bin/answer.py?hl=it&answer=99170>
- Intervista a due ingegneri di Google che hanno lavorato al progetto Google Rich Snippets: <http://radar.oreilly.com/2009/05/google-adds-microformat-parson.html>

## Conclusioni

Il formato microdata è la proposta di HTML5 per la creazione di dati strutturati che possano essere interpretati automaticamente. La sintassi è semplice: appena quattro attributi ci permettono di definire meglio i dati pubblicati. Certo, il processo di annotazione delle informazioni resta laborioso, soprattutto per la scarsità di strumenti di

supporto. È possibile che nel futuro prossimo vengano sviluppati software da terze parti per WordPress, Joomla e altri content management system, in grado di rendere meno onerosa questa attività. Abbiamo visto come Google Rich Snippets rappresenti, già oggi, un esempio concreto di impiego del formato microdata: un progetto da non trascurare, visto l'impatto positivo che un estratto più ricco delle nostre pagine è capace di generare sulle nostre pagine web per via del maggior numero di visite indotte.

## Yahoo SearchMonkey

Yahoo SearchMonkey, la piattaforma proposta da Yahoo per la gestione dei dati strutturati (un progetto per molti versi simile a quello, illustrato in queste pagine, portato avanti da Google), nonostante il buon riscontro tra il pubblico degli sviluppatori è stato chiuso il 1 ottobre 2010. Il servizio, che pure non ha mai supportato microdata ma solo i formati Microformats e RDFa, sarà profondamente rivisto a seguito del processo di integrazione che vede coinvolte Yahoo! e Microsoft nella cosiddetta "Search Alliance".

Nessun formato si è ancora imposto per l'annotazione dei dati strutturati: attualmente, comunque, il più diffuso resta Microformats; RDFa è uno strumento molto potente ma altrettanto complesso; microdata è appena nato, anche se abbiamo visto come l'investimento di conoscenze necessario per iniziare a contrassegnare le informazioni con questo formato sia davvero modesto. Purtroppo sta accadendo che, nonostante la grande attenzione che riceve nel suo complesso HTML5, sono ancora troppo pochi coloro i quali sottolineano l'importanza di questa parte della specifica. Se è ragionevole aspettarsi nei prossimi mesi un incremento di dati strutturati sul Web, è ancora difficile dire quale sarà, in definitiva, il formato che avrà la meglio.

## Capitolo 8

# Applicazioni web offline

Il campo di fruizione più immediato per le applicazioni web offline, ossia applicazioni che possono fare a meno di una costante connessione alla Rete, è quello che si rivolge in tutto o in parte al mondo dei dispositivi mobili. GMail e Calendar da più di un anno sfruttano questa tecnologia per consentire ai propri utenti di leggere mail e consultare eventi dal proprio dispositivo mobile, senza la necessità di una connessione costantemente disponibile. Inoltre, è esperienza comune perdere la connessione con il proprio smartphone, trovarsi in aree con una pessima copertura o semplicemente dover fare a meno della Rete per la durata di un viaggio aereo.

Tuttavia, applicazioni web fruibili anche parzialmente senza un costante accesso alla rete possono trovare interessanti risvolti pratici anche in un contesto molto diverso dal nostro quotidiano; si pensi per esempio alle vaste aree dei paesi emergenti o in via di sviluppo, in cui una pessima infrastruttura porta ad avere connessioni lente e spesso "a intermittenza".

Un'altra area in cui impiegare questa funzionalità è quella dei siti web che ospitano principalmente documentazione. L'autocitazione è imbarazzante, ma penso che in questo contesto possa avere senso. Quando nel 1998 ho pubblicato una guida su HTML, in una sezione del sito avevo aggiunto un file zip che conteneva tutte le lezioni da cui era composta la guida, al fine di consentire una fruizione dei contenuti non vincolata alla disponibilità di una connessione (Figura 8.1). Oggi lo stesso scopo si raggiunge in modo molto più efficace e semplice per l'utente finale, che non deve scaricare esplicitamente alcuna risorsa. La gestione delle risorse che saranno utilizzabili indipendentemente dalla presenza di una connessione a Internet avviene dietro le quinte, in modo del tutto trasparente.



**FIGURA 8.1** Quello che oggi sarebbe un metodo improbabile per la fruizione di risorse offline. Anche i colori erano improbabili, lo so!

**TABELLA 8.1** Supporto delle applicazioni web offline

ELEMENTO SUPPORTATO	BROWSER
Applicazioni web offline	Chrome 4+, Firefox 3.5+, Opera 10.6+, Safari 4+

## Supporto

Il test di supporto della funzionalità offline è molto semplice, sia con sia senza Modernizr. Prendiamo in esame entrambe le opzioni.

### **LISTATO 8.1** Test di supporto con Modernizr

```
if (Modernizr.applicationcache) {  
    // supporto esistente per le applicazioni offline  
} else {  
    // supporto inesistente!  
}
```

Il test "fai da te" senza ricorso a librerie esterne si risolve altrettanto felicemente con:

### **LISTATO 8.2** Test di supporto senza Modernizr

```
if (!window.applicationCache) {  
    // supporto esistente per le applicazioni offline  
} else {  
    // supporto inesistente!  
}
```

### NOTA

Attenzione: non si tratta di un errore di stampa: `“applicationcache”` si scrive tutto minuscolo se fate riferimento alla proprietà di Modernizr: la sintassi corretta è `Modernizr.applicationcache`. La stessa proprietà dell'oggetto `window` presenta invece la "C" maiuscola, dunque sarà sintatticamente corretto scrivere: `window.applicationCache`.

## Creare un'applicazione offline: il file manifest

Il primo passo da compiere per realizzare un'applicazione offline consiste nel creare un file manifesto che dia indicazioni sulle risorse che saranno disponibili una volta che l'accesso alla Rete non sarà più alla portata dell'utente. Questo file deve avere estensione `.manifest`.

## LISTATO 8.3 Contenuto del file scrivo.manifest

```
CACHE MANIFEST
```

```
# vers. 0.1
```

```
scrivo.html  
/css/scrivo.css  
/js/scrivo.js
```

Per quanto essenziale, questo è un file manifesto valido. La prima riga deve sempre essere composta dalle due parole `CACHE MANIFEST`. Sulle righe successive possiamo trovare una qualunque delle seguenti opzioni.

- Una riga vuota.
- Un commento. I commenti sono composti dal simbolo cancelletto `#` seguito da un testo e devono sempre apparire su una riga per conto proprio. Il motivo di questa espressa indicazione è rimuovere all'origine l'ambiguità che potrebbe sorgere se il commento si posizionasse sulla stessa riga di un indirizzo web; in tal caso potrebbe erroneamente essere scambiato per una parte integrante dell'indirizzo.
- Un titolo di sezione. Nell'esempio minimalista di file manifesto non appare alcuna intestazione di sezione; quando ciò accade si assume implicitamente che la sezione predefinita sia `CACHE:`; vedremo più avanti quali sono i tipi di sezioni che è possibile specificare in un file manifesto.
- I dati. La modalità di rappresentazione di questi dati varia da sezione a sezione. Nel nostro esempio la sola sezione da cui è composto il file `manifest` si chiama `CACHE:`; essa deve comprendere un elenco di risorse che corrisponde ai file di cui dobbiamo disporre anche in assenza di connessione.

Per avere un'idea più precisa di quali informazioni sia possibile specificare nell'ambito del file `manifest` occorre approfondire quali sono e a che cosa servono le altre sezioni: `NETWORK:` e `FALLBACK:`. La prima intestazione stabilisce che le risorse elencate nel suo ambito debbano essere esclusivamente accessibili online. Invece l'intestazione `FALLBACK:` indica per ciascuna risorsa elencata un'alternativa da utilizzare nel caso di Rete non disponibile.

Vediamo dunque come potrebbe presentarsi un file `manifest` più articolato di quello iniziale.

## LISTATO 8.4 Contenuto del file scrivo.manifest vers. 0.2

```
CACHE MANIFEST
```

```
# vers. 0.2
```

```
CACHE:  
./scrivo.html  
/css/scrivo.css  
http://www.example.org/js/scrivo.js
```

```
NETWORK:  
aggiorna.php  
FALLBACK:  
temporeale.php offline.html  
/news/* avviso.html
```

Alcune considerazioni su questa struttura prima di proseguire: nella sezione `CACHE`, la cosiddetta sezione esplicita, troviamo un elenco di risorse:

- la prima, `scrivo.html`, rappresenta un percorso relativo. In altri termini, il file `./scrivo.html` è da cercarsi a partire dalla directory corrente, che nel nostro caso è la stessa directory in cui si trova il file `manifest`. Scrivere `./scrivo.html` o `scrivo.html` è la stessa cosa. Il punto si legge: "directory corrente".
- La seconda riga, `/css/scrivo.css`, esprime un percorso assoluto. Ciò significa che il file `scrivo.css` si trova all'indirizzo `http://www.example.org/css/scrivo.css`.
- Il terzo file, `scrivo.js`, è indicato per mezzo di un indirizzo web.

Sottolineiamo tutto ciò per evidenziare il grado di libertà di cui godete nello specificare il riferimento alle risorse. Nella realtà quotidiana, a meno che non abbiate buoni motivi per comportarvi diversamente, cercate di optare per una delle tre soluzioni in modo coerente.

Si noti come la sintassi per le sezioni `CACHE` e `NETWORK` sia identica: l'unica peculiarità riguarda la sezione `FALLBACK`, dove per ciascuna riga sono indicate due risorse distinte.

La prima rappresenta quella disponibile in caso di possibile accesso alla rete, la seconda rappresenta invece l'alternativa di cui disporre in caso di offline. Si tratta insomma di una relazione uno-a-uno: per ogni risorsa specificata a sinistra, ci si rifà a quella di destra in assenza di connessione.

La seconda riga esprime una relazione uno-a-molti e si può leggere così: qualunque risorsa disponibile sotto la directory `news` sarà sostituita dal file `avviso.html` in caso di Rete non disponibile.

Una volta scritto il file `manifest`, è necessario creare una relazione con la pagina web cui si riferisce. Per ottenere questo risultato si usa il nuovo attributo `manifest` dell'elemento `<html>`, come illustrato nel seguente listato.

#### LISTATO 8.5 Un estratto del file `scrivo.html`

```
<html manifest="scrivo.manifest">  
[omissis]  
</html>
```

Sebbene la pagina `scrivo.html` sia presente tra quelle indicate nel file manifest come necessaria a garantire l'usabilità dell'applicazione anche in assenza di connessione, questa pagina verrebbe comunque conservata in locale per il semplice fatto di presentare una relazione con il file `manifest` attraverso l'omonimo attributo, e ciò anche senza una sua esplicita menzione.

## <html> tag opzionale

Fa una certa impressione, ma l'elemento `<html>`, la radice di una struttura ad albero HTML, il padre di tutti gli altri elementi, lui, proprio lui, è un tag opzionale! Ciò significa che possiamo costruire una pagina HTML perfettamente valida pur senza specificarlo. Tuttavia, se vogliamo avvalerci delle funzionalità offline offerte da questa sezione della specifica, indipendentemente da quali siano le nostre abitudini di sviluppo, dobbiamo sempre inserire questo marcatore all'inizio del documento.

Per fare in modo che il file manifest possa compiere il miracolo di un'applicazione web che non ha bisogno della Rete, per funzionare dobbiamo assicurarci che questo file sia "servito" con il MIME-type corretto.

A questo punto si rende necessaria una breve divagazione sul significato del MIME-type e su come impostarne uno, almeno su server Apache.

Per gli incalliti delle sigle, MIME sta per Multipurpose Internet Mail Extensions, ossia estensioni multiuso del servizio di posta in Internet. Infatti, il MIME-type descrive il tipo di contenuto che riceviamo via mail o che viene servito da un web server. Il suo scopo è di dare al software di posta o al browser e, più in generale, a qualunque interfaccia tra l'utente e le applicazioni della Rete, un'idea di come debbano essere processati e visualizzati i dati ricevuti. Più precisamente, questa informazione viene fornita al browser attraverso un'intestazione Content-type, di cui vediamo di seguito il formato:

### **LISTATO 8.6** Formato di un'intestazione Content-type

Content-type: tipo/sottotipo

Il `tipo` indica la macrocategoria di appartenenza del dato, mentre il `sottotipo` ne rappresenta il formato specifico. Alcuni esempi di tipi di contenuto molto diffusi in Rete sono:

- `text/html` per i documenti HTML;
- `text/css` per i fogli di stile;
- `text/plain` per file di puro testo che non saranno interpretati;
- `image/png` per immagini che utilizzano il formato Portable Network Graphics;

- `image/jpg`: le foto sono nella maggior parte dei casi servite con questo Content-type;
- `application/pdf` per i documenti PDF.

Il file manifesto che abbiamo creato, pur essendo un puro file di testo, non può vedersi attribuito il tipo `text/plain` perché altrimenti il browser rischierebbe di non impiegarlo in modo corretto. Dobbiamo istruire il server a consegnare il file manifest con un suo specifico tipo, che sarà il `text/cache-manifest`.

Modificare la lista dei tipi di file serviti da un web server è una tipica attività di configurazione svolta dall'amministratore del server. Se l'amministratore siete voi e state utilizzando un server Apache, allora il vostro compito sarà di modificare il file `.htaccess` aggiungendo la direttiva `AddType` come proposto di seguito.

#### **LISTATO 8.7** Direttiva AddType da aggiungere nel file `.htaccess`

---

`AddType: text/cache-manifest .manifest`

#### **NOTA**

---

Il punto prima dell'estensione è facoltativo e l'estensione è sensibile alla combinazione di caratteri maiuscoli e minuscoli; abbiate perciò cura di applicare a questo tipo di file un'estensione che sia consistente con quanto dichiarato.

Per capire come funziona in concreto la gestione di un applicativo web offline osserviamo nel dettaglio che cosa accade al verificarsi degli scenari più frequenti.

#### **Scenario 1: prima visita di `scrivo.html`**

Se il browser visita una pagina web che dichiara l'esistenza di un file di tipo manifest per la prima volta, significa che non esistono copie più vecchie di questo file sulla macchina dell'utente. Il browser scarica e interpreta il file manifest e inoltra richiesta al server per ciascuna delle risorse che devono essere utilizzabili in assenza di connessione.

Le fasi salienti di questo processo sono demarcate da eventi specifici.

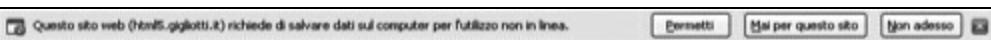
- `checking`: è sempre il primo evento a essere scatenato e ciò accade ogni volta si analizzi il file manifest per capire se debba o meno essere scaricato.
- `downloading`: in questo caso l'evento è scatenato perché non esiste una precedente versione del file manifest.
- `progress`: evento scatenato per ogni risorsa da scaricare in locale.
- `cached`: tutte le risorse menzionate nel file manifest sono state scaricate e l'applicazione è integralmente disponibile in locale sulla macchina dell'utente.

Prima di dare inizio a questo processo il browser può proporre una notifica all'utente per renderlo consapevole di quanto sta accadendo (Figura 8.2). Si tratta di un'opzione di configurazione, per cui è possibile bloccare del tutto l'utilizzo non in linea di dati da qualunque sito o, al contrario, rendere possibile lo scaricamento di dati in locale senza avvisare di quanto sta accadendo. La barra di notifica è una soluzione intermedia proposta da alcuni browser (Firefox e Opera).

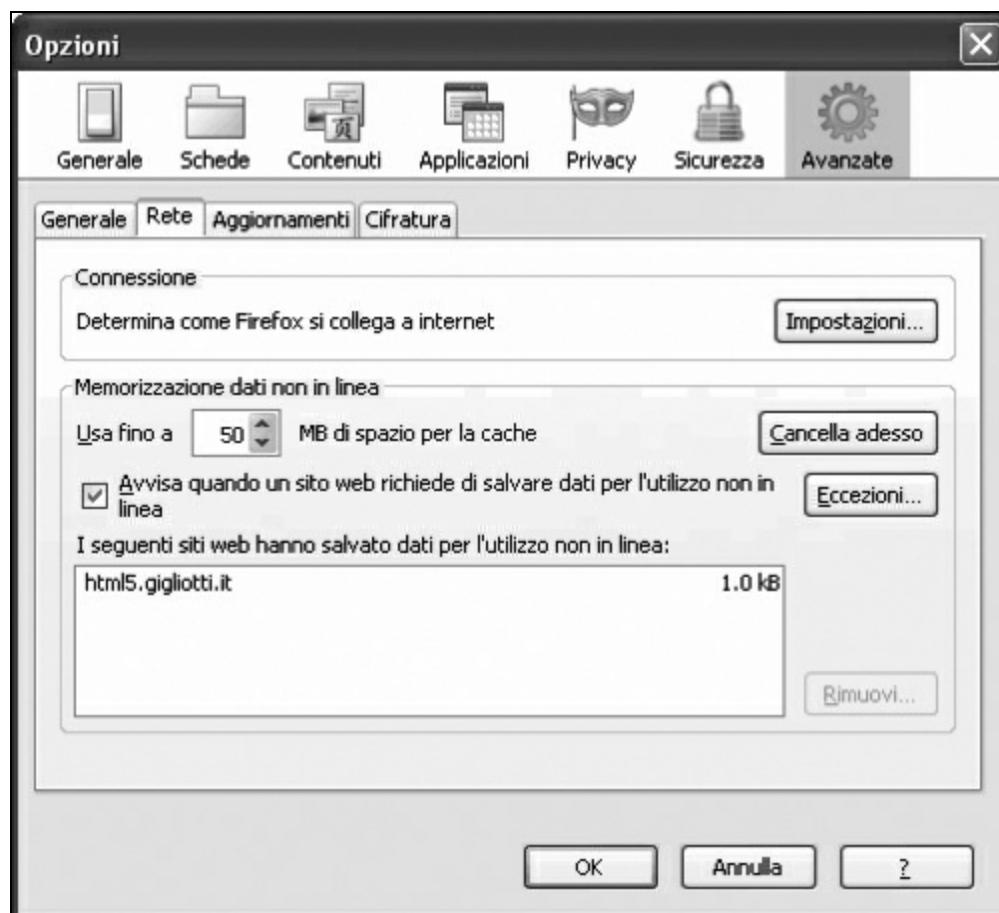
Dal pannello di configurazione è possibile visualizzare quale sito ha salvato dei dati per una consultazione non in linea e quanto spazio occupa ciascuno di essi (Figura 8.3).

## Scenario 2: visite successive alla pagina web scrivo.html. Nessuna modifica al file manifest

Il browser visita una pagina web che dichiara l'esistenza di un file di tipo manifest, quindi determina se sono state apportate modifiche rispetto alla versione precedente. Poiché le versioni sono identiche, non accade nulla. Gli eventi di questo processo sono:



**FIGURA 8.2** Firefox avvisa l'utente della richiesta, scatenata dal file manifest, di scaricare dati per un utilizzo non in linea.



**FIGURA 8.3** L'utente è messo a conoscenza del sito web che ha salvato dati per l'utilizzo non in linea, oltre alla dimensione di tali dati.

- `checking`: indipendentemente dal fatto che il file manifest venga scaricato o meno, questo evento viene sempre scatenato, e sarà sempre il primo evento della sequenza;
- `noupdate`: poiché dal confronto tra la copia del file manifest conservata in locale e quella presente in remoto non sono emerse differenze, non accade nulla. Per segnalare questo stato di cose è scatenato l'evento `noupdate`.

## Scenario 3: visite successive alla pagina web `scrivo.html`. Il file manifest risulta modificato

Il browser visita una pagina web che dichiara l'esistenza di un file di tipo manifest e identifica l'esistenza di differenze tra la versione conservata in locale e quella che risiede in remoto. Avvia quindi lo scaricamento in locale di tutte le risorse elencate nel file.

Gli eventi che si susseguono in questo scenario sono i seguenti:

- `checking`: il solo punto fermo di tutti gli scenari è l'evento `checking`, sempre presente e sempre il primo nella sequenza;
- `dowloading`: in questo caso l'evento è scatenato perché non esiste una precedente versione del file manifest;
- `progress`: evento scatenato per ogni risorsa da scaricare in locale;
- `updateready`: a differenza di quanto abbiamo visto nello scenario 1, l'ultimo evento di questa sequenza è `updateready`. Esso segnala che tutte le risorse presenti nel file manifest sono state scaricate un'altra volta.

### NOTA

`updateready` ci comunica che l'aggiornamento delle risorse ha avuto luogo con successo; tuttavia questi dati non sono ancora disponibili all'utente. Per rendere effettive queste modifiche anche agli occhi dell'utente dovremo utilizzare la funzione `swapCache()` che, come suggerisce il nome stesso, sostituisce la vecchia versione con quella nuova appena scaricata.

## Scenario 4: una o più risorse tra quelle segnalate nel file manifest non sono disponibili

La premessa è la stessa dello scenario precedente. A fronte di una differenza tra versione locale e versione remota del file manifest si dà avvio allo scaricamento dei file. Tuttavia, in questo caso la sequenza di eventi cambia:

- `checking`: è sempre il primo evento a essere scatenato e ciò accade ogni volta si analizzi il file manifest per capire se debba o meno essere scaricato;
- `downloading`: in questo caso l'evento è scatenato perché non esiste una precedente

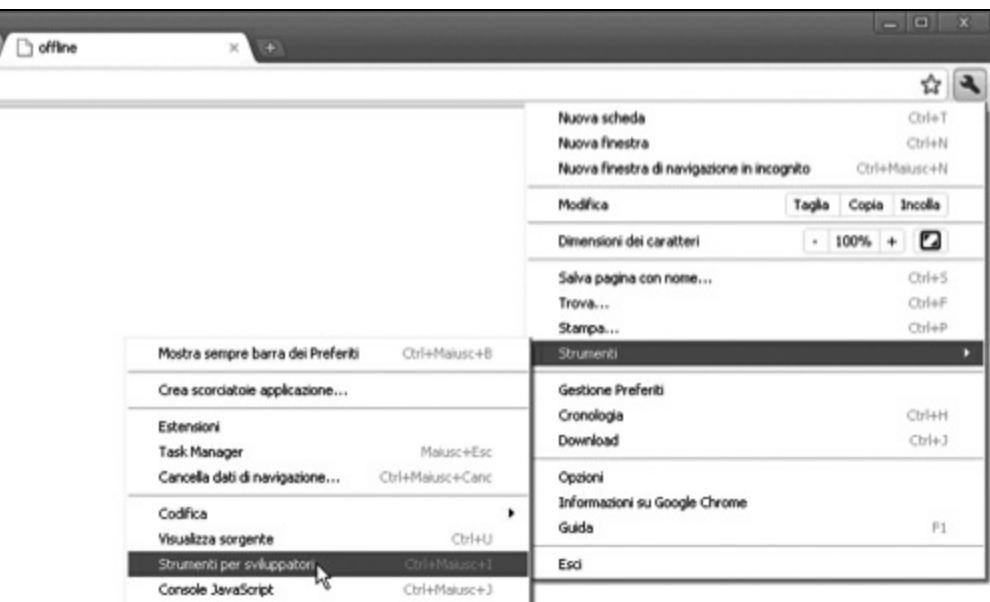
versione del file manifest;

- `progress`: evento scatenato per ogni risorsa da scaricare in locale;
- `error`: l'impossibilità di ottenere una copia di una delle risorse elencate nel file manifest determina l'evento `error`.

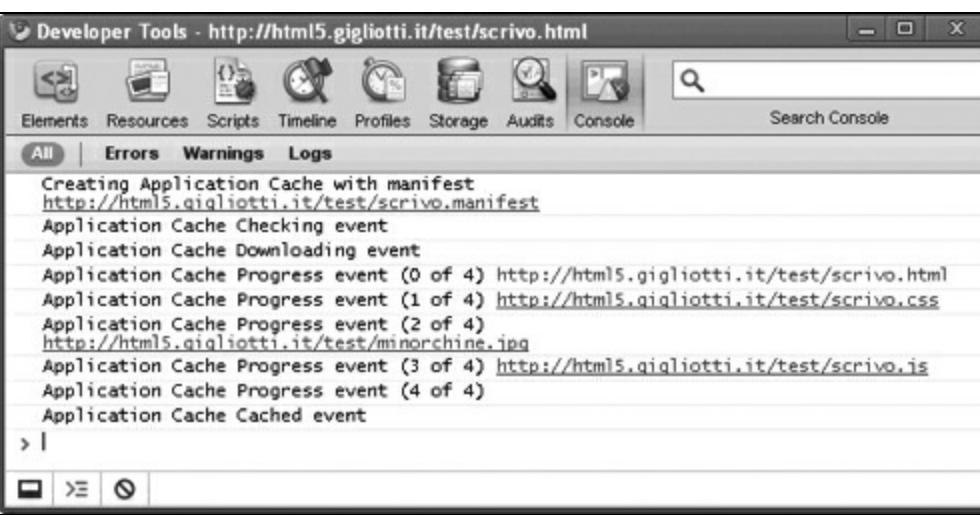
Altri casi possono essere all'origine di un errore: l'impossibilità di recuperare lo stesso file manifest o la pagina web che presenta il riferimento al file manifest e infine il caso, tanto più probabile quanto più alto è il traffico verso il sito, del file manifest modificato in concomitanza con il processo di scaricamento dei dati. Queste e altre insidie legate alle normali modalità di funzionamento del protocollo HTTP rendono più complesso il debugging, ossia l'attività di analisi volta a identificare e rimuovere gli errori.

## Debugging: strumenti utili

Durante la realizzazione di applicazioni offline e, in seguito, proprio nella più delicata fase di debugging può risultare utile avvalersi degli "strumenti per sviluppatori" (i developer tools) di Chrome (Figura 8.4) che, pur con una spartana interfaccia a linea di comando, permettono di analizzare il comportamento dell'applicativo attraverso gli eventi che ne contrassegnano il ciclo di vita (Figura 8.5). In aggiunta, la sezione Storage offre un ulteriore e più approfondito livello di dettaglio, consentendo di affiancare alla sequenza cronologica degli eventi anche l'URL e la dimensione di ciascuna risorsa oggetto dell'applicativo offline (Figura 8.6).



**FIGURA 8.4** Come accedere agli strumenti per sviluppatori di Chrome.



**FIGURA 8.5** L'interfaccia a linea di comando di Chrome.

## Riferimenti alle risorse citate e altri link utili

- La specifica ufficiale per le applicazioni offline: <http://www.w3.org/TR/html5/offline.html>
- Un articolo che spiega il significato del MIME-type e l'effetto che un'errata definizione dello stesso produce sui browser: [https://developer.mozilla.org/en/Properly\\_Configuring\\_Server\\_MIME\\_Types](https://developer.mozilla.org/en/Properly_Configuring_Server_MIME_Types)

The screenshot shows the Chrome Developer Tools interface with the 'Storage' tab selected. The 'APPLICATION CACHE' section is expanded, showing the contents of the 'scrivo' cache. The table lists the following resources:

Resource	Type	Size
http://html5.gigliotti.it/test/minorchine.jpg	Explicit	53.16KB
http://html5.gigliotti.it/test/scrivo.css	Explicit	357B
http://html5.gigliotti.it/test/scrivo.html	Master Explicit	1.01KB
http://html5.gigliotti.it/test/scrivo.js	Explicit	373B
http://html5.gigliotti.it/test/scrivo.manifest	Manifest	413B

Below the table, the logs section shows the same Application Cache creation and progress events as in Figure 8.5.

**FIGURA 8.6** La sezione Storage degli strumenti per sviluppatori di Chrome.

- La guida ufficiale su come configurare un MIME-type su server Apache: [http://httpd.apache.org/docs/current/mod/mod\\_mime.html](http://httpd.apache.org/docs/current/mod/mod_mime.html)
- La documentazione pubblicata dal Mozilla Developer Network sulle risorse offline: [https://developer.mozilla.org/En/Offline\\_resources\\_in\\_Firefox](https://developer.mozilla.org/En/Offline_resources_in_Firefox)
- Un interessante articolo, corredata da video, che illustra come utilizzare in gli eventi scatenati dalle applicazioni offline: <http://www.bennadel.com/blog/2029-Using-HTML5-Offline-Application-Cache-Events- In-Javascript.htm>

## Conclusioni

Le applicazioni web disponibili in modalità non in linea porteranno nuova linfa creativa al Web. Alcuni siti potrebbero persino cambiare la propria filosofia e ripensarsi attorno all'idea di un set minimo di servizi disponibili sempre e comunque, ai quali potrebbero aggiungersi funzionalità ulteriori con l'accesso alla Rete. La mancata implementazione da parte di Internet Explorer 9 e alcune difficoltà di debugging rappresentano attualmente gli unici punti deboli della tecnologia.

# Oltre i cookie: più spazio per i tuoi dati!

Dal 1994 fino a oggi i dati relativi alle applicazioni web, o almeno una piccola parte di questi, è stata archiviata direttamente sulle macchine degli utenti di tali applicazioni: tutto ciò è stato possibile grazie ai cookie. Una parte di HTML5 ha rielaborato questa esperienza più che decennale. Mirando al superamento di alcune soluzioni che non hanno trovato il successo sperato, perché basate su implementazioni proprietarie (lo userData di Microsoft) o perché richiedevano un apposito componente aggiuntivo (è il caso dei cosiddetti flash script), HTML5 propone un'interfaccia di programmazione implementata in modo nativo e condivisa tra i browser, denominata Web Storage. Si tratta di un'area di applicazioni molto importante vista l'ubiquità dei cookie nelle applicazioni web odierne. La Tabella 9.1 ne illustra il grado di supporto presso i principali browser sul mercato.

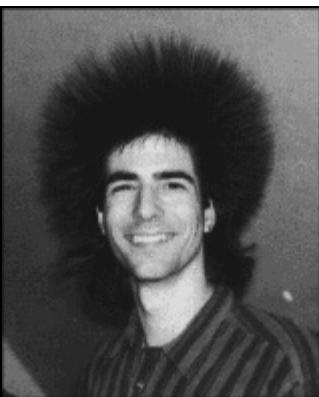
**TABELLA 9.1** Supporto per l'interfaccia di programmazione sul Web Storage

ELEMENTO SUPPORTATO	BROWSER
Web Storage API	Chrome 4+, Firefox 3.0+, IE8+, Opera 10.5+, Safari 4+

Poiché il Web Storage si propone come soluzione più robusta ed efficace dei cookie, ipotecandone così la progressiva sostituzione, iniziamo il capitolo proprio con una breve introduzione su questi ultimi.

## Introduzione essenziale ai cookie

I cookie sono stati inventati intorno alla metà degli anni Novanta del secolo scorso (gulp!) da Lou Montulli (Figura 9.1) quando lavorava presso Netscape; sono definiti nella RFC 2109.



**FIGURA 9.1** Lou Montulli, creatore dei cookie nonché del famigerato tag <blink>.

## Request for Comments

La definizione data da Wikipedia per RFC aiuta a chiarire la funzione di questi documenti: "Una Request for Comments (termine inglese che tradotto letteralmente in italiano significa 'richiesta di commenti'), più nota con la sigla RFC, è un documento che riporta informazioni o specifiche riguardanti nuove ricerche, innovazioni e metodologie dell'ambito informatico o, più nello specifico, di Internet".

Un cookie, termine che in italiano significa "biscotto", è un file di testo di piccole dimensioni, il cui peso non può superare i 4 KB (questa e altre limitazioni sono indicate in Tabella 9.2). Il suo contenuto è composto da coppie chiave-valore, e ogni coppia è separata dalla successiva da un carattere di punto e virgola ", ".

Il cookie viene utilizzato per completare il processo di autenticazione sui più svariati servizi online; in generale, proprio l'autenticazione, oltre al tracciamento degli oggetti riposti nel carrello della spesa (paradigma usato da quasi tutti i negozi online) e al poco gradevole tracciamento delle abitudini degli utenti, è il suo impiego più diffuso.

I cookie vengono creati dal server (il computer remoto sul quale risiede il sito web che si sta consultando) o via JavaScript e sono salvati in locale sulla macchina del visitatore. Successivamente alla sua creazione, il browser si preoccuperà di re-inoltrarlo al server a ogni richiesta HTTP (tipicamente, ogni richiesta di accesso a una nuova pagina o di download di una risorsa ecc.). Proprio il continuo trasferimento del file da e verso il server consente di mantenere uno stato tra più richieste e dunque di tracciare queste richieste per gli scopi sopra indicati.

**TABELLA 9.2** Limitazioni dei cookie come da specifica RFC-2109

TIPO DI LIMITAZIONE	VALORE
Dimensione del file cookie	4096 byte ossia 4 KB
Numero massimo di cookie consentiti	300
Numero massimo di cookie consentiti (provenienti dallo stesso dominio)	20

Nel Listato 9.1 è mostrato un esempio di una risposta HTTP ottenuta da Google a seguito di una ricerca nella quale si può notare la riga di intestazione che inizia per `Set-Cookie`.

### **LISTATO 9.1** Esempio di risposta dell'intestazione HTTP

```
Cache-Control: private, max-age=0
Date: Fri, 29 Oct 2010 21:53:42 GMT
Expires: -1
Content-Type: text/html; charset=UTF-8
Set-Cookie: PREF=ID=d97d4b08ed4e84ac:FF=0:TM=1288389222:LM=1288389222:S=tf56Req0
M_5eFZ3t; expires=Sun, 28-Oct-2012 21:53:42 GMT; path=/; domain=.google.it
Content-Encoding: gzip
Transfer-Encoding: chunked
Server: gws
X-XSS-Protection: 1; mode=block
```

## Supporto

Come detto in apertura di capitolo, il Web Storage gode di un ottimo supporto tra i browser, ma ciò non toglie che è sempre buona norma testarne il supporto al fine di proporre come alternativa i ben rodati cookie.

### **LISTATO 9.2** Test di supporto con Modernizr

```
// sostituisci localstorage con sessionstorage per
// testare l'altra modalità di salvataggio dati in locale
if (Modernizr.localstorage){
    // localStorage è supportato!
} else {
    // come strategia di supporto alternativo
    // per il salvataggio dei tuoi dati puoi
    // usare direttamente il server, oppure
    // optare per l'utilizzo dei cookie
}
```

Volendo fare a meno della libreria esterna Modernizr e dovendo testare il supporto di `localStorage` e `sessionStorage`, che sono due proprietà dell'oggetto `window`, sarebbe naturale scrivere:

### **LISTATO 9.3** Test di supporto "fai da te"

```
if (window.localStorage) {
```

```
// Supporto OK
} else {
    // Funzionalità non implementata
}
```

Questa tecnica funziona su tutti i browser a eccezione di Firefox. In questo caso infatti, se i cookie sono disabilitati, o se lo storage è disattivato, ogni tentativo di accesso alle proprietà `localStorage` o `sessionStorage` dell'oggetto `window` scatenerà un errore di sicurezza di tipo `NS_ERROR_DOM_SECURITY_ERR`, per prevenire il quale è necessario utilizzare l'operatore JavaScript `in`; inoltre, si dovrà catturare l'eccezione con un costrutto `try/catch`.

Per questo motivo la stessa libreria Modernizr esegue al suo interno questo tipo di test:

#### **LISTATO 9.4** Test di supporto "fai-da-te" con gestione per Firefox

```
// per testare sessionStorage è sufficiente
// sostituirlo nell'esempio con localStorage
function testStorage() {
    try {
        return ('localStorage' in window) && window.localStorage !== null;
    } catch(e) {
        return false;
    }
}
```

#### **NOTA**

Questo è uno dei casi in cui Modernizr rende semplice il controllo di una data funzionalità proteggendoci dalle insidie che questo comporta. Man mano che il supporto di HTML5 e CSS3 si estenderà potremo gradualmente rimuovere questo codice di controllo oltre a quello posto in essere per assicurare un'adeguata strategia alternativa.

## Le due facce del Web Storage

Guardando più da vicino questa interfaccia di programmazione, scopriamo subito l'esistenza di due diverse modalità di intendere il salvataggio dei dati sul client:

`sessionStorage` e `localStorage`. Sebbene entrambe condividano la stessa modalità di accesso e di manipolazione dei dati salvati localmente (l'interfaccia di programmazione che ora vedremo è identica), hanno caratteristiche distinte che le rendono idonee a soddisfare diverse esigenze.

`sessionStorage` presenta diverse affinità con i cookie, per questo motivo torna utile schematizzarne le principali differenze con una tabella che ne metta a confronto le caratteristiche (Tabella 9.3).

#### **TABELLA 9.3** Differenze tra cookie e sessionStorage

COOKIE	SESSIONSTORAGE
Ogni cookie può raggiungere una dimensione massima di 4 KB	Lo spazio per ciascun oggetto <code>sessionStorage</code> è pari a un massimo di 5 MB (Opera consente, previa autorizzazione dell'utente, di estendere ulteriormente questa quantità di dati)
Ogni cookie è scambiato sistematicamente tra il server e il client generando così richieste più corpose	A meno di esplicita configurazione da parte dell'autore dello script, i dati non sono sistematicamente scambiati tra server e client
Nel caso di accesso da più tab/finestre del browser a uno stesso sito di commercio elettronico che richiede autenticazione basata sui cookie, possono potenzialmente prodursi effetti indesiderati sulle transazioni (per esempio acquisti doppi!), a causa della condivisione di tutti i tab/finestre dei cookie creati dal sito web.	Ogni oggetto <code>sessionStorage</code> , dunque i dati in esso contenuti, "vivono" all'interno del singolo tab/singola finestra da cui si accede al sito web. Più accessi contemporanei allo stesso sito web da tab diversi dello stesso browser risultano del tutto separati. Ogni accesso dispone della propria copia dei dati in locale.

Le modalità `localStorage` e `sessionStorage` si distinguono tra loro solo per visibilità dei dati e loro durata, come emerge dalla Tabella 9.4 di confronto.

**TABELLA 9.4** Differenze tra `localStorage` e `sessionStorage`

SESSIONSTORAGE	LOCALSTORAGE
I dati salvati localmente per mezzo di <code>sessionStorage</code> non sopravvivono al tab o alla finestra in cui sono stati creati. Alla chiusura di quest'ultima i dati sono rimossi. A questo comportamento generale fa eccezione il caso di ripristino del browser, per cui può accadere che, a fronte di una chiusura forzata o comunque anomala del browser, venga proposta l'opzione di ripristinare le sessioni precedentemente in essere (Figura 9.2)	La persistenza dei dati va oltre il ciclo di vita della finestra del browser. Alla chiusura della finestra (o del tab) i dati non sono rimossi, rimanendo così disponibili per futuri accessi, anche a distanza di giorni
I dati sono disponibili solo nell'ambito della finestra o del tab all'interno del quale sono stati generati. La loro visibilità è dunque limitata. L'apertura di un nuovo tab o di una nuova finestra determina l'inizializzazione di una nuova sessione.	La disponibilità dei dati si estende a tutte le finestre/tab del browser
Campo di applicazione: tutti quei casi in cui il processo è di breve durata e richiede il salvataggio di dati di carattere temporaneo	Campo di applicazione: utile in tutti i casi in cui l'utente può interrompere il processo per poi completarlo in un secondo momento anche a distanza di giorni (per esempio aggiungere articoli nel carrello della spesa oggi per poi completare l'acquisto tra una settimana)

# Interfaccia di programmazione

Con le idee più chiare su cosa sia e come il Web Storage si differenzia dai cookie possiamo preoccuparci di come utilizzare in concreto l'interfaccia di programmazione per il salvataggio dei dati in locale. Introduciamo dunque `setItem()` e `getItem()`, funzioni utili per accedere in scrittura e lettura a questa area locale in cui risiedono i dati. Queste funzioni servono per creare un elemento (ossia una coppia chiave-valore) e per recuperarne il valore mediante l'indicazione della chiave corrispondente. Quanto si illustra di seguito si applica anche al `localStorage`, con le fondamentali differenze sulla visibilità e persistenza dei dati illustrate in Tabella 9.4.

## **LISTATO 9.5** Salvare un dato in locale 1

```
sessionStorage.setItem("nome", "Giube");
```

`setItem()` accetta due parametri: il primo rappresenta la chiave, il secondo è il valore a essa associato. Una diversa notazione che consente di ottenere lo stesso risultato è la seguente:

## **LISTATO 9.6** Salvare un dato in locale 2

```
sessionStorage.nome = "Giube";
```

La specifica stabilisce che, mentre la chiave deve essere sempre una stringa (del testo, insomma), il valore potrebbe ospitare un numero intero o decimale o anche altri tipi di oggetti; tuttavia, in forza dell'attuale stadio di implementazione della specifica, tutti i valori sono salvati come se fossero delle stringhe.

## Salvataggio locale di oggetti

Nel caso in cui si abbia intenzione di salvare in locale dei numeri prima di utilizzarli sarà necessario fare ricorso ai metodi `parseInt()` e `parseFloat()` che restituiscono un numero partendo dalla loro "rappresentazione stringa". Se si ha comunque necessità di salvare degli oggetti, fintanto che la specifica non conoscerà, su questo punto, una migliore implementazione, sarà possibile usare JSON e la sua funzione `stringify()` per ottenere una rappresentazione stringa dell'oggetto. Successivamente, per ripristinare nuovamente la struttura originaria dell'oggetto si utilizza il metodo `parse()`.

Lo script che segue illustra come sia possibile salvare un oggetto.

```
// creo un oggetto usando la notazione JSON  
  
var oggetto = { "nome" : "Giube",
```

```

        "hobby" : "meccanicaquantistica",
        "professione" : "barista",
        "anni" : 31.5
    };

    // ne salvo in locale la rappresentazione stringa
    sessionStorage.setItem("oggString", JSON.stringify(oggetto));
    /// leggo la rappresentazione stringa conservata in locale
    var oggString = sessionStorage.getItem("oggString");
    /// converto nuovamente la stringa in oggetto
    var oggRipristinato = JSON.parse(oggString);
    // ne leggo una proprietà
    alert(oggRipristinato["anni"] + 1);

```

Adottando questa tecnica la rappresentazione stringa dell'oggetto salvato in locale con chiave `oggString` sarà:

```
{ "nome": "Giube", "hobby": "meccanicaquantistica", "professione": "barista", "anni": 31.5}
```

A dimostrazione che la conversione da stringa a oggetto è avvenuta con successo ho impiegato il numero in un'addizione senza applicare alcuna conversione esplicita con `parseFloat()`.

Una volta salvato, il valore associato a una data chiave può essere letto per mezzo di `getItem()`. La funzione accetta un argomento che rappresenta il nome della chiave di cui si vuole ottenere il valore corrispondente.

#### **LISTATO 9.7** Estrarre un dato salvato localmente 1

```
var nome = sessionStorage.getItem("nome");
```

Questa espressione di assegnamento estrae il valore associato alla chiave `"nome"` e ne attribuisce tale valore alla variabile `nome`. La notazione alternativa che produce lo stesso risultato è la seguente:

#### **LISTATO 9.8** Estrarre un dato salvato localmente 2

```
var nome = sessionStorage.nome;
```

È inoltre possibile scorrere le chiavi attraverso un uso combinato delle proprietà `length`, che restituisce il numero di chiavi esistenti, e `key(pos)`, che restituisce il nome della chiave avente la posizione `pos` nella lista di chiavi.

#### **LISTATO 9.9** Scorrere le coppie chiavi valori

```
<script>
```

```
// imposto alcune chiavi
sessionStorage.nome = "Giube";
sessionStorage.ruolo = "barista";
sessionStorage.citta = "milano";
sessionStorage.hobby = "meccanica quantistica";
var chiave, valore;
for (var i = 0; i < sessionStorage.length; i++) {
  chiave = sessionStorage.key(i);
  valore = sessionStorage.getItem(chiave);
  alert("chiave: " + chiave + " - valore: " + valore);
}
</script>
```

## NOTA

Non fate alcun affidamento in merito all'ordine in cui sono proposte le chiavi: varia da browser a browser e varia anche a fronte della modifica di uno qualunque dei valori associati alle chiavi.

Per prendere visione dei dati salvati localmente è possibile aiutarsi sia con Chrome sia con Opera. Il primo, attraverso il percorso accessibile dal suo menu Personalizza e Controlla Chrome | Strumenti | Strumenti per sviluppatori permette di visualizzare informazioni utili, quali appunto le coppie chiave-valore impostate dalla pagina web che stiamo visitando (Figura 9.3). Il secondo invece propone uno strumento analogo dal più altisonante nome DragonFly (Figura 9.4).

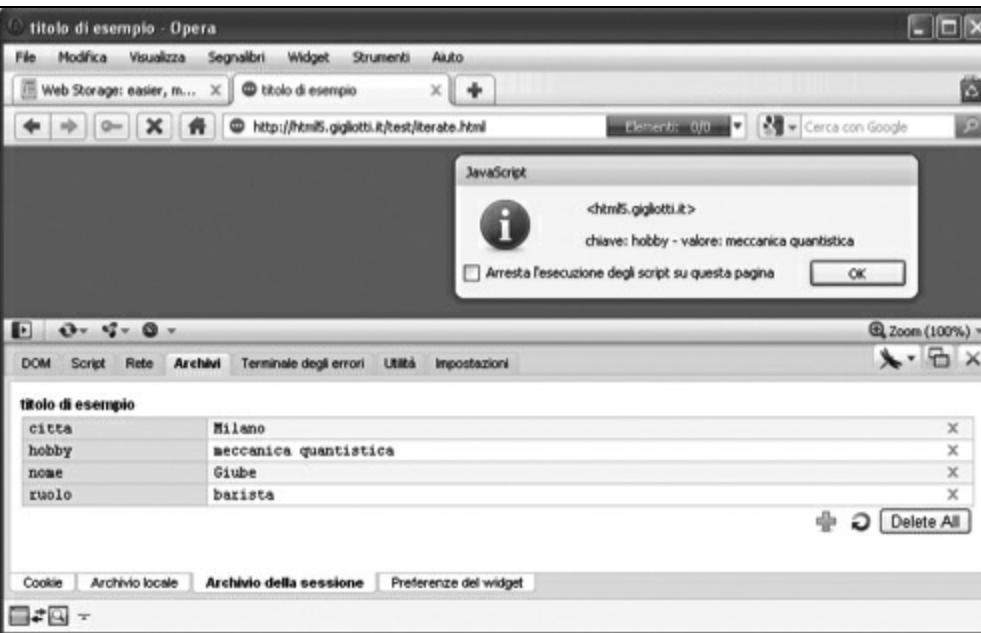


The screenshot shows the 'Storage' tab in the Chrome Developer Tools. The left sidebar lists 'LOCAL STORAGE' and 'SESSION STORAGE' under the domain 'html5.gigliotti.it'. The main panel displays a table with two columns: 'Key' and 'Value'. The data is as follows:

Key	Value
nome	Giube
ruolo	barista
colore	rosso
citta	milano

**FIGURA 9.3** Tra gli strumenti per sviluppatori proposti da Chrome è presente anche un pannello di visualizzazione tabellare dei dati salvati localmente.

Qualora si renda necessario rimuovere un particolare elemento, dunque una coppia chiave-valore, o addirittura tutti i dati presenti in locale, si utilizzeranno rispettivamente `removeItem()` e `clear()`. La prima funzione accetta come argomento il nome della chiave da eliminare (unitamente all'elemento che contrassegna) mentre la seconda, come suggerisce lo stesso nome, rimuove integralmente tutti i dati presenti.



**FIGURA 9.4** Opera DragonFly, un valido strumento di supporto per lo sviluppo.

## Come accorgersi di modifiche dei dati salvati in locale

Nei casi più semplici, sapere come leggere e scrivere i dati sulla macchina dell’utente può essere sufficiente. Tuttavia, come può capitare per le applicazioni più complesse, occorre anche tenere traccia di eventuali modifiche poste in essere sui dati. Per soddisfare questa esigenza l’interfaccia di programmazione mette a disposizione un evento, chiamato `storage`, notificato ogni qualvolta i dati sono modificati. Non è sufficiente invocare una qualunque delle funzioni che possono modificare i dati, la modifica deve essere reale. In altri termini, questo evento non viene notificato quando si verifica uno qualunque dei seguenti casi.

- La funzione `clear()` è invocata quando non esistono dati. Poiché non esiste nulla, non ci sarà nulla da cancellare.
- La funzione `removeitem()` è invocata con riferimento a una chiave inesistente. In tal caso, poiché la chiave non esiste, nulla verrà rimosso dai dati conservati localmente.
- La funzione `setItem()` è utilizzata per assegnare a una chiave già esistente lo stesso identico valore rispetto a quello attuale.

Nell’ipotesi, invece, di una concreta modifica dei dati, l’evento `storage` offre l’accesso, in sola lettura, a una serie di proprietà che ci aiutano a capire quale modifica abbia avuto luogo:

- `key`: la chiave oggetto di modifica;
- `oldValue`: il valore associato alla chiave prima della modifica;
- `newValue`: il valore associato alla chiave dopo la modifica;
- `url`: l’indirizzo del documento la cui chiave ha subito la modifica;

- **storageArea**: un riferimento all'oggetto che rappresenta l'insieme dei dati che, nel loro complesso, hanno subito la modifica.

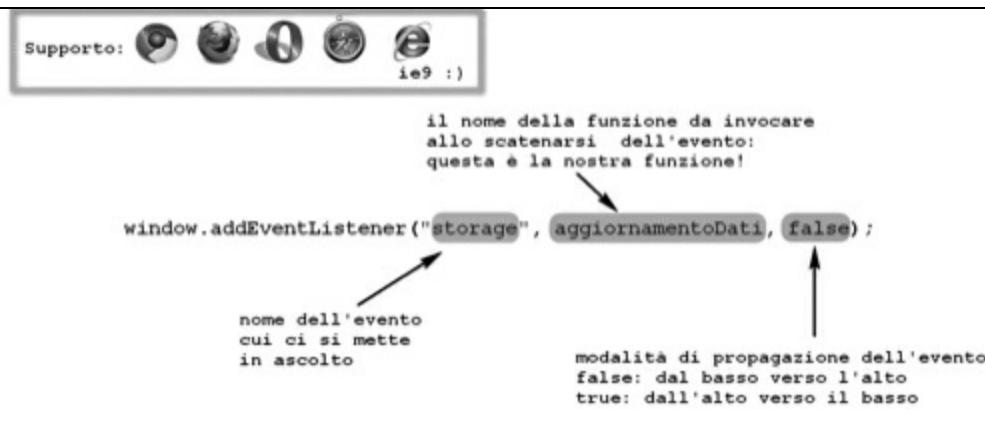
Ed ecco un esempio che illustra come mettersi in ascolto dell'evento per catturarne le relative informazioni:

#### LISTATO 9.10 In ascolto dell'evento storage

```
if (window.addEventListener) {
  window.addEventListener("storage", aggiornamentoDati, false);
} else if (window.attachEvent) { // per IE
  window.attachEvent("onstorage", aggiornamentoDati);
}

function aggiornamentoDati(event) {
  var storageEvent = event || window.event;
  alert("chiave modificata: " + storageEvent.key);
}
```

È possibile mettere una propria funzione in ascolto di un dato evento, così che allo scatenarsi di tale evento la funzione possa essere chiamata in causa e svolgere il compito assegnatole. Si ottiene questo risultato per mezzo di `addEventListener()` che, come suggerisce il nome stesso, non fa altro che aggiungere un "ascoltatore" a un dato evento.



**FIGURA 9.5** `addEventListener()`: mettersi in ascolto di un evento.

`addEventListener()` però non è riconosciuta dalle versioni di Internet Explorer precedenti alla 9. L'istruzione `if else` serve per adottare il modello alternativo di ascolto degli eventi che si basa su `attachEvent()`, necessario per Internet Explorer 6, 7 e 8.

Si aggiunge un ascoltatore di eventi, così da poter ricevere una notifica nel momento in cui è scatenato l'evento `storage`. Quando ciò accade, è eseguita la funzione `aggiornamentoDati()`. Essa ottiene un riferimento alle proprietà dell'evento `storage` per mezzo dell'argomento `event`. Tuttavia, ancora una volta, per ottenere un codice che funzioni anche con Internet

Explorer scriviamo:



**FIGURA 9.6** attachEvent(): mettersi in ascolto di un evento (con Internet Explorer).

#### **LISTATO 9.11** Ottenere un riferimento all'evento

```
var storageEvent = event || window.event;
```

Ciò può leggersi come: se la funzione `aggiornamentoDati()` non ha ottenuto un riferimento all'evento appena scatenato, allora recupera questo riferimento dall'oggetto `window`.

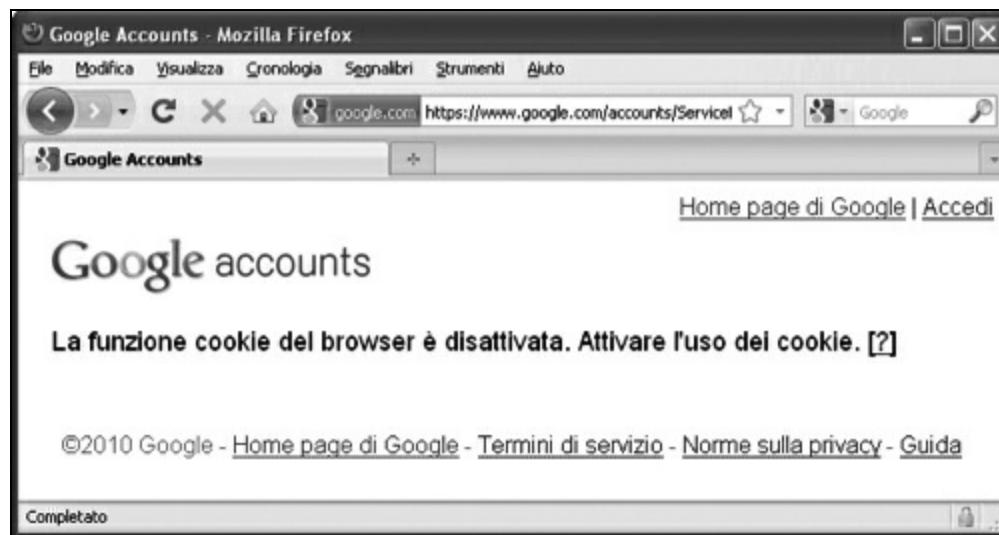
#### **ATTENZIONE**

Il supporto all'evento `storage` presenta ancora lacune importanti sui principali browser. Il test di esecuzione è andato a buon fine solo su Opera 10.62.

## Tutela dei dati personali

I cookie sono stati visti in passato come una minaccia alla privacy, e ancora oggi rimane alto il numero di persone che decide di disabilitare questo strumento preferendo così rinunciare del tutto ai servizi che ne prevedono l'uso come condizione imprescindibile per il loro corretto funzionamento. Per chi usa la Rete, disabilitare i cookie è una scelta radicale con effetti importanti. Solo per citare alcuni siti tra i più visitati al mondo: tutti i servizi di Google che richiedono l'autenticazione diventano inaccessibili. Non siete in grado di leggere la posta su GMail o condividere documenti con Google Documents, solo per citare due tra i servizi più utili proposti dal colosso di Mountain View (Figura 9.7). Non potete pubblicare le vostre foto su Flickr, la famosa comunità per la condivisione di fotografie (Figura 9.8), né far sapere al mondo quanti caffè avete bevuto oggi usando Twitter, piattaforma di micro-blogging che proprio non concepisce l'idea si possano disabilitare, tant'è che non si preoccupa di proporre un messaggio per invitare all'abilitazione degli stessi: semplicemente ricarica la pagina di autenticazione come se nulla fosse accaduto!

Il Web Storage, proprio perché rispetto ai cookie si profila come strumento più duttile ed efficace, ne amplifica le potenziali minacce ai dati personali.



**FIGURA 9.7** Anche in Google l'autenticazione non ha luogo senza cookie.



**FIGURA 9.8** Il motivo per cui il browser "rifiuta" di autenticarsi sono i cookie disabilitati.

A questo proposito è illuminante l'esempio riportato dall'editore della specifica in merito a un uso fraudolento dello strumento per ottenere un sofisticato sistema di tracciamento dell'utente: "Un inserzionista terza parte (o qualunque entità in grado di distribuire i propri contenuti su più siti) potrebbe utilizzare un identificativo univoco salvato nell'area di storage locale per tracciare un utente su sessioni multiple, costruendo così un profilo degli interessi dell'utente che consenta di predisporre una forma di pubblicità altamente mirata. In associazione con un sito che sia consapevole della reale identità dell'utente (per esempio un sito di commercio elettronico che richieda di autenticare le credenziali), questo potrebbe mettere gruppi oppressivi nelle condizioni di prendere di mira singoli individui con più grande precisione di quanto non sia possibile in un mondo in cui esista un Web puramente anonimo".

# Riferimenti alle risorse citate e altri link utili

- Ultima versione stabile pubblicata dal W3C: <http://www.w3.org/TR/webstorage/>
- Ultima bozza della specifica sul Web Storage pubblicata presso il sito del W3C: <http://dev.w3.org/html5/webstorage/>
- Definizione di RFC data da Wikipedia: [http://it.wikipedia.org/wiki/Request\\_for\\_Comments](http://it.wikipedia.org/wiki/Request_for_Comments)
- Il documento Request For Comments 2109 che regolamenta i cookie: <http://www.ietf.org/rfc/rfc2109.txt>
- Il Mozilla Developer Center propone un articolo sul DOM Storage (un altro modo di chiamare il Web Storage). Da qui troverete link ad alcune applicazioni di esempio e relativi tutorial: <https://developer.mozilla.org/en/dom/storage>
- L'equivalente del Mozilla Developer Center per il mondo Opera si chiama Dev.Opera e propone un interessante articolo sul Web Storage: <http://dev.opera.com/articles/view/web-storage/>
- Profilo di Lou Montulli su Wikipedia: [http://en.wikipedia.org/wiki/Lou\\_Montulli](http://en.wikipedia.org/wiki/Lou_Montulli)
- La foto di Lou Montulli: <http://www.nndb.com/people/977/000030887/>

## Conclusioni

L'interfaccia di programmazione Web Storage nasce dall'esigenza di superare le idiosincrasie legate ai cookie. E, in effetti, questa specifica rappresenta già oggi una splendida realtà: il supporto è ottimo, la dimensione di spazio disponibile per il salvataggio dei dati è di almeno un ordine di grandezza maggiore se confrontata con i cookie. Infine, le due modalità di salvataggio dei dati: `sessionStorage` e `localStorage`, se applicate agli scenari per cui sono concepite (si veda l'ultima riga della Tabella 9.4), costituiscono un tassello imprescindibile per tutte le nuove applicazioni web. Il Web Storage, oltre alla possibilità di sviluppare applicazioni che non soffrono di temporanee interruzioni di connessione alla Rete, permetterà di irrobustire applicazioni esistenti (scrivere una mail piuttosto che un commento a un articolo letto sul proprio quotidiano preferito) e sarà linfa vitale per applicazioni completamente nuove, in grado di sfruttare la Rete anche a fronte di una connessione disponibile solo a singhiozzo. Rimangono invece vive le preoccupazioni relative a un uso scorretto legato al tracciamento degli utenti e delle loro abitudini da parte di aziende senza scrupoli, che adottino tecniche di profilazione aggressive per poi rivendere a terzi informazioni straordinariamente dettagliate.

# Capitolo 10

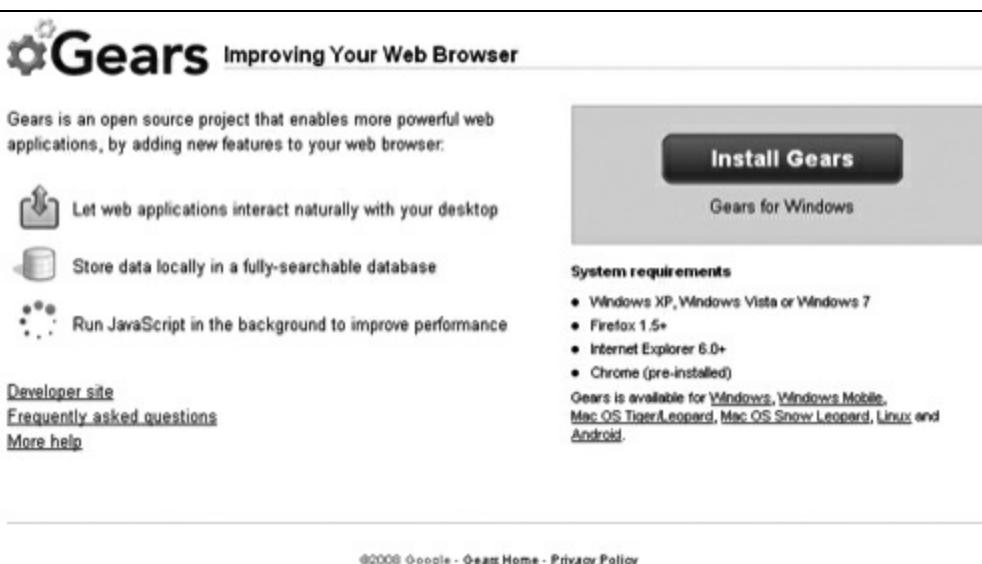
# Geolocalizzazione

Avviso: il capitolo che state per leggere non ha molto a che fare con il titolo di questo manuale! Sì, perché la localizzazione geografica (da qui in avanti semplicemente geolocalizzazione), ossia la possibilità di identificare in modo più o meno accurato la vostra posizione geografica, non è oggi, né è mai stata in passato, parte della specifica HTML5. Il motivo per cui se ne discute in questo manuale è che si tratta di una tecnologia legata al Web sviluppata in concomitanza con HTML5 e impropriamente associata a HTML5 persino in articoli di stampa specialistica. Di certo, questa interfaccia di programmazione rappresenta un altro elemento di quel puzzle che saranno le applicazioni web di prossima generazione, e rende ancora più rovente il tema, già caldo, della tutela dei dati personali sul Web.

**TABELLA 10.1** Supporto alla localizzazione geografica

ARGOMENTO	BROWSER
Geolocation API	Chrome 5, Firefox 3.5+, Opera 10.6, Safari 5

L'interfaccia di programmazione relativa alla geolocalizzazione può essere facilmente estesa anche al browser Internet Explorer fino alla versione 6 grazie al componente aggiuntivo realizzato da Google, che prende il nome di Gears ed è disponibile all'indirizzo: <http://gears.google.com> (Google Chrome implementa il componente Gears in modo nativo).



The screenshot shows the Gears project website. At the top, there's a logo with a gear and the text "Gears Improving Your Web Browser". Below the logo, a text box says: "Gears is an open source project that enables more powerful web applications, by adding new features to your web browser." To the right, a large button says "Install Gears" with the subtext "Gears for Windows". Below this, there's a "System requirements" section with the following list:

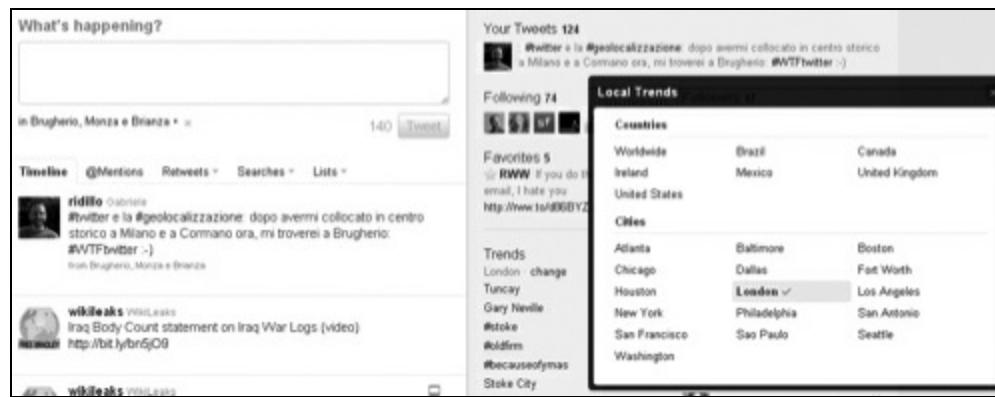
- Windows XP, Windows Vista or Windows 7
- Firefox 1.5+
- Internet Explorer 6.0+
- Chrome (pre-installed)

At the bottom of the page, there are links for "Developer site", "Frequently asked questions", and "More help".

**FIGURA 10.1** Google Gears estende la geolocalizzazione anche a vecchie versioni di Internet Explorer (dalla versione 6 in avanti).

## Fratello, dove sei?

I motivi che possono spingere a chiedere informazioni in merito alle coordinate geografiche dei nostri utenti sono i più svariati ma, a ben vedere, rientrano sempre in uno di questi due casi: si vuole arricchire un servizio già offerto integrando con le coordinate geografiche altri dati di cui già si dispone, oppure si intende prestare un nuovo servizio in cui la localizzazione dell'utente è conditio sine qua non per l'erogazione dello stesso. Alla prima categoria appartiene l'utilizzo della geolocalizzazione come avviene presso Twitter che, su permesso dell'utente, può tracciare la località dell'autore di un twit, il singolo messaggio di 140 caratteri di questa piattaforma di micro-blogging. In questo modo Twitter è in grado di proporre a tutti i suoi utenti alcuni Trends: un breve elenco dei temi più discussi in una data città (Figura 10.2).



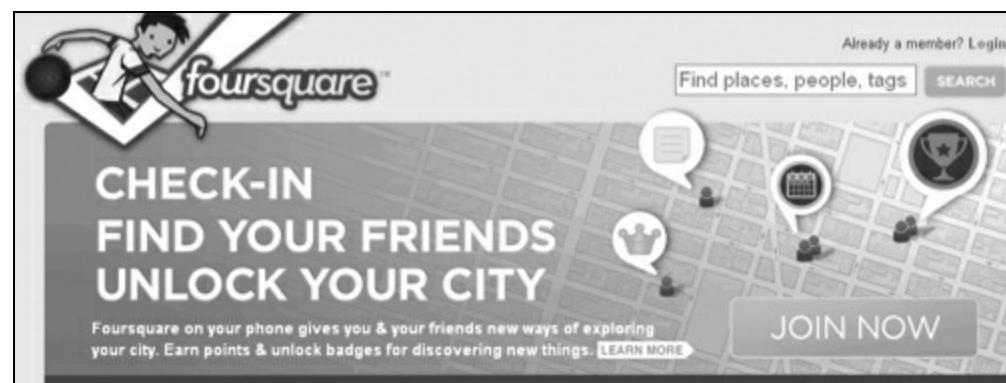
**FIGURA 10.2** Messaggi geolocalizzati su Twitter, piattaforma di micro-blogging.

### Grado di precisione

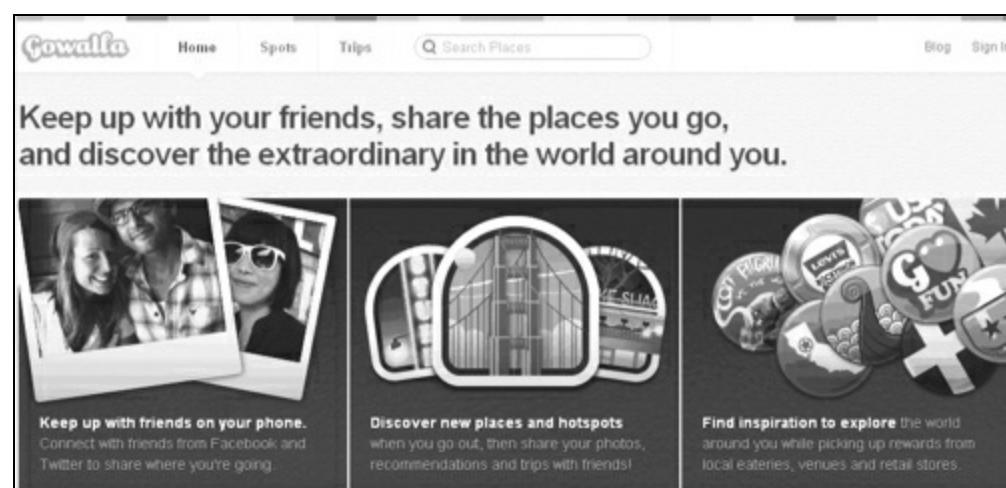
Alla stato attuale va rilevato che il grado di precisione della geolocalizzazione applicata da Twitter presenta un esaltante grado di libertà, se è vero che nell'arco di tre giorni, pur collegandomi sempre dalla mia abitazione, mi ha apparentemente localizzato in tre diversi comuni appartenenti a due province diverse: "Centro storico, Milano", "Cormano" e "Brugherio, Monza Brianza", sbagliando in quest'ultimo caso di circa 20 km. È giusto comunque evidenziare che si tratta di una tolleranza accettabile, dato che il servizio serve semplicemente a raggruppare i messaggi identificando i temi più dibattuti nell'ambito di una certa area geografica. Va da sé che la stessa tolleranza nella localizzazione di un utente non è accettabile se il servizio è quello di un navigatore satellitare che deve guidarci con ben altra precisione lungo un dato percorso. L'accuratezza con cui la longitudine e la latitudine sono rilevate dipende dalla modalità stessa di rilevazione. Se la rilevazione basata sull'indirizzo IP risulta potenzialmente poco precisa, potendoci collocare anche a parecchi chilometri di distanza da dove ci troviamo realmente, la geolocalizzazione basata su GPS (acronimo per Global Positioning System) è di gran lunga più accurata.

In altri casi la geolocalizzazione non è un servizio accessorio, ma condizione indispensabile per erogare il servizio stesso. Foursquare, Gowalla e Google Latitude (Figure 10.3, 10.4 e 10.5) sono nomi che potrebbero non dirvi nulla se non possedete uno smartphone; in caso contrario è probabile siano tra le prime applicazioni che avete

installato. Si tratta di social media, come Facebook, ma con una peculiarità: tutti questi servizi si basano sulla condivisione delle coordinate geografiche dell’utente che, in questo modo, mette a conoscenza gli amici della sua posizione geografica per essere raggiunto più facilmente. Alcune attività commerciali sfruttano il check-in dell’utente, ossia la sua auto-segnalazione presso i locali dell’attività stessa con la quale può ottenere uno sconto o un riconoscimento di qualche altro tipo (Foursquare assegna il titolo di “sindaco” all’utente che in un certo periodo si è dimostrato più assiduo frequentatore di un certo locale).



**FIGURA 10.3** Foursquare si basa sulla geolocalizzazione via GPS dei suoi utenti.



**FIGURA 10.4** Gowalla è un altro famoso servizio di geolocalizzazione dei propri utenti.

On a mobile device? Visit <http://m.google.com/latitude>

### See where your friends are right now.

Google Latitude lets you stay close with your friends from your phone, computer, or both. [Learn more](#) or [Watch video](#)

- Find your friends on a map — see nearby friends and meet up.
- Share where you are with the friends you choose.
- Control your location and privacy.

### Do more with your location using Latitude apps

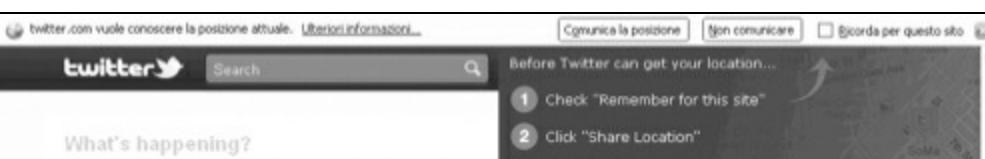


Developers: Learn about the [Google Latitude API](#)

**FIGURA 10.5** Google Latitude: la geolocalizzazione secondo la grande G.

## Cenni sulla sicurezza e sul trattamento dei dati personali

È evidente che mentre alcuni potrebbero manifestare un legittimo e inconfondibile entusiasmo anche solo di fronte alla possibilità di comunicare ad amici e parenti di aver appena raggiunto il casello di Balocco lungo la A4, o di essere assiduo frequentatore di locali di tendenza, altri penseranno al Grande Fratello di orwelliana memoria e rabbividiranno all'idea di lasciare che informazioni così frequenti e dettagliate sui propri spostamenti siano esposte con tanta semplicità. Per questo motivo, come vedremo, nessuna comunicazione delle coordinate geografiche in cui si trova il dispositivo che usiamo per collegarci (ossia, in definitiva, dove ci troviamo noi) potrà avere luogo se non dietro nostro esplicito consenso: le Figure 10.6, 10.7, 10.8, 10.9 e 10.10 illustrano come i principali browser in commercio abbiano fatto propria questa direttiva espressamente prevista dalla specifica. Inoltre, l'interfaccia utente del browser deve sempre consentire di revocare il privilegio precedentemente accordato alla comunicazione di tali dati.



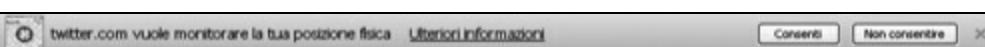
**FIGURA 10.6** Twitter mostra di avere con Firefox 3.6 una integrazione più forte presentando una grafica che enfatizza la barra di notifica.



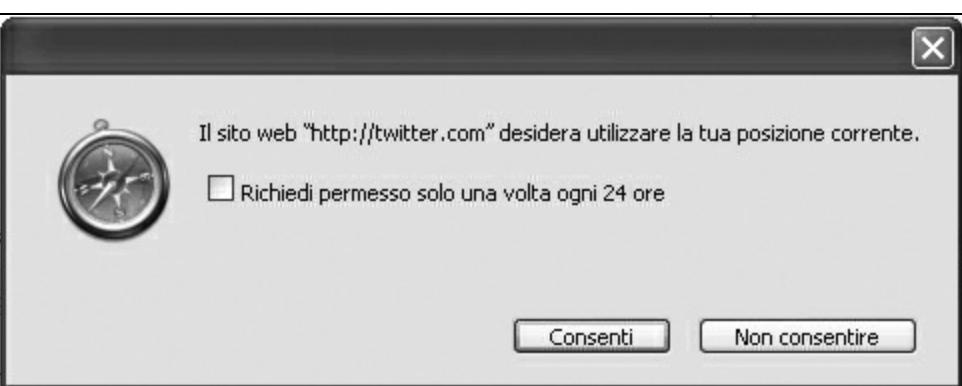
**FIGURA 10.7** Nell'ultima beta disponibile di Firefox muta la modalità di avviso.



**FIGURA 10.8** La barra di notifica proposta da Opera.



**FIGURA 10.9** La barra di notifica proposta da Chrome.

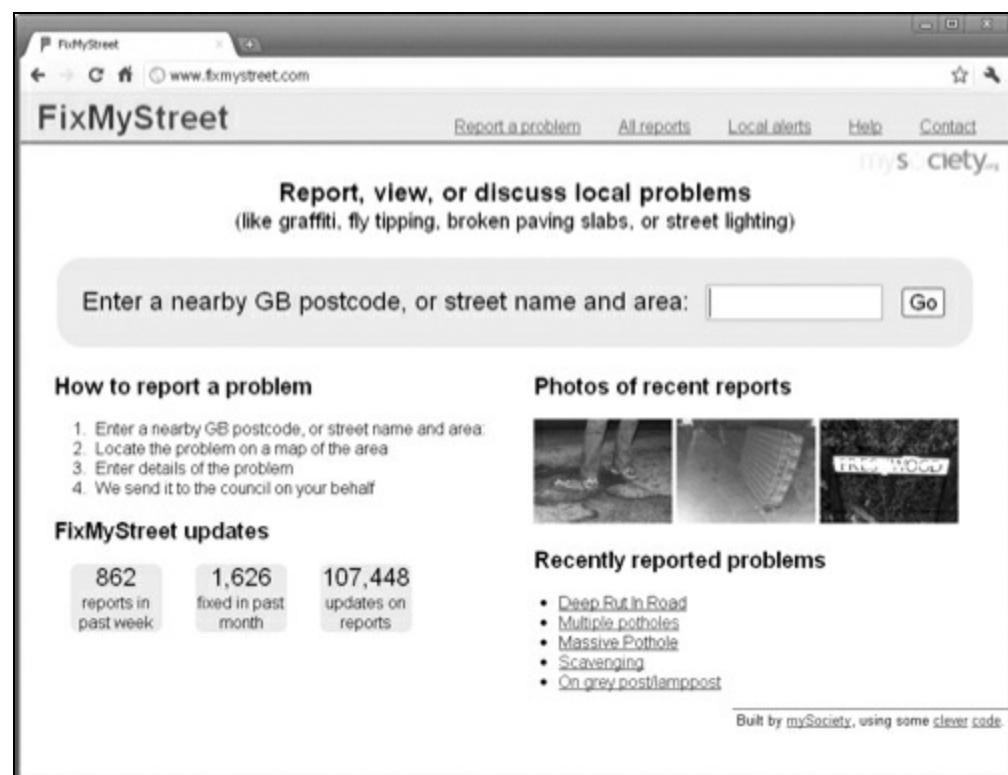


**FIGURA 10.10** Safari opta per una finestra modale decisamente più visibile, in cui chiede anche di reiterare l'identificazione delle coordinate geografiche una volta al giorno.

## Geolocalizzazione e Open Data

Al di là della declinazione sul versante dei social network, esistono applicazioni di questa tecnologia decisamente meno modaiole ma con risvolti che possono essere anche di impegno civico e sociale. Infatti, è proprio l'uso incrociato di dati geografici, demografici e di servizi pubblici disponibili sul Web in una forma riutilizzabile che rende possibile lo sviluppo di servizi prima d'ora inimmaginabili. L'iniziativa britannica Show Us A Better Way (<http://www.showusabetterway.co.uk>) è illuminante in questo senso. Si tratta di una competizione in cui ci si sfida alla ricerca di un impiego utile dei dati resi pubblici dall'amministrazione britannica a ogni livello. Nelle precedenti edizioni di questa competizione i dati geospatiali sono stati quelli usati in modo più intenso e hanno

prodotto le applicazioni più disparate. <http://www.fixmystreet.com> (Figura 10.11) offre un esempio da manuale. È un sito in cui cittadini riportano disservizi come dissesti del manto stradale, illuminazione pubblica non funzionante e simili, il tutto corredato da foto a documentazione del problema e posizionamento dello stesso prontamente disponibile sulle mappe di Google: le segnalazioni sono rintracciabili per codice postale, via, città!



**FIGURA 10.11** FixMyStreet: un uso intelligente di dati geospatiali forniti dagli utenti.

## Supporto alla geolocalizzazione

Anche in questo caso, Modernizr si rivela la soluzione più immediata per testare il supporto della Geolocation API.

### LISTATO 10.1 Testare il supporto alla Geolocation API con Modernizr

```
<script>
  if (Modernizr.geolocation) {
    // Geolocalizzazione supportata
  } else {
    // invitare l'utente a inserire in modalità manuale
    // un riferimento alla propria posizione geografica
    // come per esempio il codice di avviamento postale.
  }
</script>
```

In questo caso, a differenza di altri scenari, l'opzione "fai da te", senza ricorso alla libreria esterna, risulta ugualmente semplice: consiste nel testare l'esistenza dell'oggetto `navigator.geolocation`.

### **LISTATO 10.2** Testare il supporto alla Geolocation API con soluzione "fai da te"

```
<script>
  if (!!navigator.geolocation){
    // Geolocalizzazione supportata
  }
</script>
```

L'operatore `!!` restituisce `true` se l'oggetto esiste e `false` se l'oggetto è `undefined`, ossia inesistente.

Adesso che sappiamo come assicurarsi che quest'interfaccia di programmazione sia effettivamente utilizzabile, mettiamola alla prova con un primo rudimentale ma efficace esempio di determinazione delle coordinate geografiche.

### **LISTATO 10.3** Determinazione di longitudine e latitudine

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Geolocalizzami: longitudine e latitudine</title>
    <script src="modernizr-1.5.min.js"></script>
    <style>
      body {
        margin: 1.8em;
        font: 1.7em/2em verdana, sans-serif;
      }
      span { color: #f00; }
      li { list-style-type:none; }
    </style>
  </head>
  <body>
    <script>
      if (Modernizr.geolocation) {
        navigator.geolocation.getCurrentPosition(geolocalizzami);
      } else {
        alert("geolocalizzazione non supportata dal browser!");
      }
      function geolocalizzami(position) {
```

```

document.getElementById("lon").innerHTML = position.coords.longitude;
document.getElementById("lat").innerHTML = position.coords.latitude;
document.getElementById("when").innerHTML = new Date(position.timestamp);
}

</script>
<ul>
<li>Longitudine: <span id="lon">-</span></li>
<li>Latitudine: <span id="lat">-</span></li>
<li>Coordinate identificate in (data . ora): <span id="when">-</span></li>
</ul>
</body>
</html>

```

## Il codice e la Geolocation API

Questo esempio ci fornisce lo spunto per passare in rassegna le funzioni e gli oggetti che nel loro insieme rappresentano l'interfaccia di programmazione di geolocalizzazione.

Dapprima si esegue il test del supporto delle funzioni JavaScript di cui si farà uso, il tutto per mezzo di Modernizr. Se il browser non riconosce gli oggetti con cui vogliamo operare, viene proposta all'utente una finestra modale per informarlo che il proprio browser non supporta questa funzionalità.

### Finestra modale

Un tipico esempio di finestra modale in JavaScript (Figura 10.12) è dato da:

```
alert("messaggio in finestra modale");
```

Si tratta di una finestra che non consente all'utente di eseguire altre attività sino alla sua dismissione (nell'esempio sarà sufficiente premere il pulsante OK per chiudere la finestra e riprendere l'operatività ordinaria).



**FIGURA 10.12** Esempio di finestra modale in JavaScript.

Se il browser supporta il set di funzioni JavaScript relative alla geolocalizzazione verrà eseguita la funzione `getCurrentPosition()` appartenente all'oggetto `geolocation`. Non appena invocata, questa, a sua volta, lancerà (in modalità asincrona) una richiesta di identificazione geografica, quindi l'esecuzione del codice procederà immediatamente. In questo caso termina perché non esistono altre dichiarazioni all'interno del ramo `if`.

## L'importanza di un'esecuzione asincrona

Il fatto che il tentativo di determinazione delle coordinate avvenga in modalità asincrona non è un dettaglio tecnico: è il modo fondamentale per garantire una gradevole esperienza all'utente. Se il tempo necessario per determinare tali coordinate fosse molto lungo (parecchi secondi), l'operatività dell'utente risulterebbe bloccata per tutto questo periodo.

Se il tentativo di recupero delle informazioni geografiche va a buon fine, sarà eseguita la funzione `geolocalizzami()`, il cui nome è proposto come unico argomento di `getCurrentPosition()`. Ma facciamo un passo indietro. Non appena si scatena il tentativo di localizzazione del visitatore, o meglio del dispositivo da cui il suo browser sta accedendo alla nostra pagina web, il browser interrompe subito il processo in attesa di un'esplicita autorizzazione a procedere. Questo accade per mezzo di barre di notifica, finestre modali o altri elementi grafici come illustrato nelle Figure dalla 10.6 alla 10.10. Solo in caso di risposta affermativa il tentativo di tracciamento geografico ha effettivamente luogo.

L'esecuzione della funzione `geolocalizzami()` avviene dunque se, e solo se, l'utente ha esplicitamente acconsentito al tracciamento e se questo si è concluso con successo (Figura 10.13).

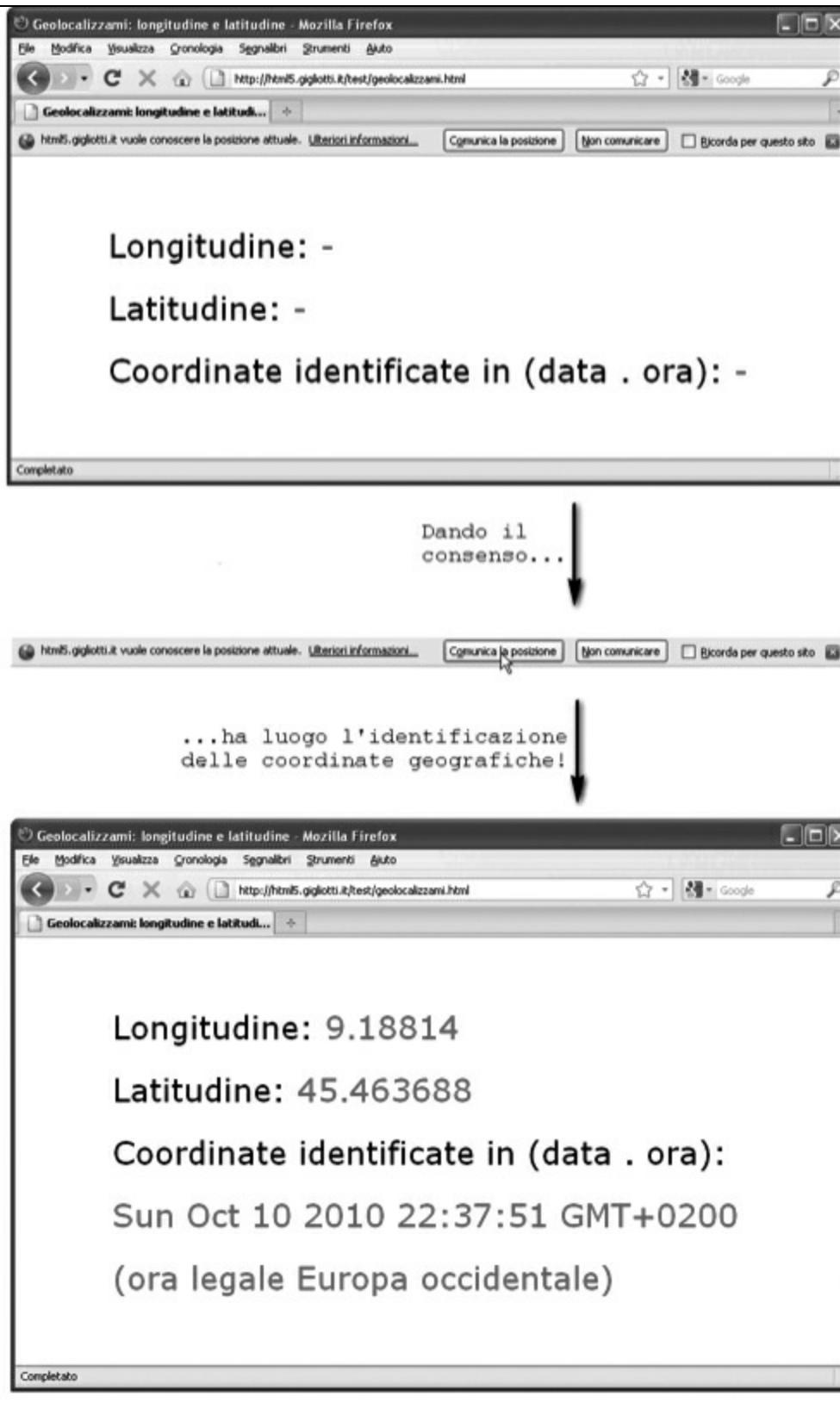
In questo scenario le informazioni utili a identificare la locazione geografica dell'utente risiedono nell'oggetto di tipo `Position` cui la funzione `geolocalizzami()` ottiene un riferimento. Questo oggetto è pre-popolato con i tanto agognati dati sulla longitudine e latitudine, oltre alla data/ora di avvenuta identificazione di tali informazioni. Le coordinate sono proprietà dell'oggetto `coords`, a sua volta proprietà dell'oggetto `position`, dunque accessibili per mezzo della sintassi:

### **LISTATO 10.4** Longitudine e latitudine sono proprietà dell'oggetto coords

```
position.coords.longitude;  
position.coords.latitude;
```

Longitudine e latitudine non sono le uniche coordinate accessibili per mezzo dell'oggetto `coords`. Ne esistono altre cinque, elencate di seguito.

- `coords.altitude`. Se il dispositivo su cui è installato il browser non riesce a identificare l'altitudine, il valore di questo attributo sarà pari a `null`.
- `coords.accuracy`. Longitudine e latitudine possono essere identificate con gradi di precisione diversi a seconda del sistema adottato per tracciarle. Questa proprietà indica un valore dell'approssimazione di tali coordinate, espresso in metri. Il valore numerico sarà reale non negativo.



**FIGURA 10.13** Identificazione di longitudine e latitudine e data/ora di identificazione.

- `coords.accuracy`. Esprime in metri il livello di precisione dell'altitudine. Se l'altitudine ha valore `null`, anche questa proprietà sarà pari a `null`.
- `coords.heading`. Indica la direzione espressa in gradi determinata partendo dal nord geografico e procedendo in senso orario. Se il dispositivo è stazionario il suo valore sarà `NaN` (acronimo di Not a Number).
- `coords.speed`. È la velocità espressa in metri per secondo. Se la velocità è pari a

zero, la direzione sarà pari a `NaN`.

timestamp, invece, è una proprietà diretta dell'oggetto `position`, perciò accessibile come descritto di seguito:

#### **LISTATO 10.5** Data e ora di tracciamento delle coordinate geografiche (espressa in millisecondi)

```
position.timestamp
```

timestamp restituisce il numero di millisecondi che intercorrono dalla mezzanotte del 01/01/1970 al momento esatto in cui queste coordinate sono state tracciate. Per tradurre questo numero in una data che non sia comprensibile solo alla macchina, usiamo il costruttore `new Date(millisecondi)` che, senza ulteriori formattazioni della data, produce una stringa decisamente più comprensibile come quella illustrata in Figura 10.13.

### 01/01/1970 00:00:00 (UTC)

Per convenzione, nei sistemi Unix il tempo viene calcolato come differenza in millisecondi rispetto alla mezzanotte del primo gennaio 1970 nel fuso orario UTC (Tempo Coordinato Universale). Lo stesso sistema è stato adottato, tra gli altri, anche dal linguaggio di scripting del Web, JavaScript. Perché un sistema così cervellotico? Perché, come illustra bene la pagina web di Wikipedia Italia sul tema specifico ([http://it.wikipedia.org/wiki/Tempo\\_\(Unix\)](http://it.wikipedia.org/wiki/Tempo_(Unix))), "questo tipo di rappresentazione, oltre a essere compatta, è indipendente dai fusi orari, ed è quindi direttamente confrontabile anche tra calcolatori situati a grandi distanze geografiche tra loro, ed evita di dover effettuare aggiustamenti nel caso, per esempio, di dati trasmessi da un fuso orario all'altro".

Una curiosità: il 9 settembre 2001 alle 03:46:40 CET (ora dell'Europa centrale) il numero dei millisecondi trascorsi fino a quel momento dalla mezzanotte del primo gennaio 1970 è stato esattamente pari a 1 miliardo: evento persino festeggiato da alcuni gruppi informatici in tutto il mondo: decisamente una delle pochissime cose che si siano festeggiate quel giorno!

### Rilevamento “one-shot” delle coordinate geografiche con `getCurrentPosition()`

Osserviamo nel dettaglio la firma della funzione `getCurrentPosition()` per scoprire quanto duttile essa sia in realtà.

#### **LISTATO 10.6** Longitudine e latitudine sono proprietà dell'oggetto `coords`

```
void getCurrentPosition(in PositionCallback successCallback,  
                        in optional PositionErrorCallback errorCallback,  
                        in optional PositionOptions options);
```

Questa è la funzione da invocare nel caso in cui si sia interessati a una lettura unica e

non ripetuta delle coordinate geografiche dell'utente. Se volessimo invece monitorare le variazioni e dunque tracciare lo spostamento dell'utente nello spazio, dovremmo impiegare il metodo `watchPosition()` illustrato più avanti.

Scopriamo subito che gli argomenti di `getCurrentPosition()` possono essere uno, due oppure tre: mentre il primo è obbligatorio, il secondo e il terzo sono in effetti opzionali.

- L'argomento `PositionCallback successCallback` esprime un riferimento alla funzione che gestisce l'invocazione asincrona in caso di successo (consenso dell'utente al tracciamento della propria posizione e conseguente positiva determinazione delle coordinate geografiche). In parole povere, se tutto va per il meglio, il primo argomento di `getCurrentPosition()` corrisponde al nome della funzione che verrà eseguita. A questa funzione verrà passato un riferimento all'oggetto `Position`, le cui proprietà sono state poc'anzi elencate.
- L'argomento `PositionErrorCallback errorCallback` esprime un riferimento alla funzione la cui esecuzione è scatenata in caso di diniego, espresso dall'utente, di farsi localizzare geograficamente o, se il tentativo di localizzazione è invece autorizzato, in caso di un qualunque altro errore che abbia impedito la determinazione delle coordinate geografiche. A questa funzione verrà passato un riferimento all'oggetto `PositionError`, su cui torneremo più avanti.
- Il terzo argomento è un oggetto di tipo `PositionOptions` le cui tre proprietà `enableHighAccuracy`, `timeout` e `maximumAge` permettono di intervenire sulle modalità con cui ha luogo il processo di tracciamento. Vedremo in che modo, modificando il codice di esempio.

Possiamo dunque sfruttare queste conoscenze per rivedere il listato sopra indicato e scrivere una funzione che gestisca il caso di fallimento aggiungendo un argomento di `errorCallback`, nonché per configurare alcuni aspetti del tracciamento.

#### **LISTATO 10.7** Secondo argomento: funzione da eseguire in caso di errore

```
navigator.geolocation.getCurrentPosition(geolocalizzami, geoErrore);
```

`geoErrore()` è la funzione che gestisce il caso di errore.

#### **LISTATO 10.8** La funzione geoErrore()

```
function geoErrore(positionError) {  
  alert(positionError.message + " codice[" + positionError.code + "]);  
}
```

`geoErrore()` è il punto in cui costruire la strategia alternativa, che può consistere nell'invitare l'utente a inserire il proprio codice d'avviamento postale o di "auto-

localizzarsi" su una mappa con un semplice "punta-e-clicca". Nel caso di errore si ottiene inoltre il riferimento a un oggetto di tipo `PositionError`. Attraverso le sue proprietà `message` e `code`, abbiamo accesso a un messaggio di errore potenzialmente utile a fini di debug. Allo stesso modo, la proprietà `code` propone un valore numerico che indica le possibili cause dell'errore, sintetizzate in Tabella 10.2.

Il terzo argomento della funzione `getCurrentPosition()` è un oggetto di tipo `PositionOption` che ci consente di configurare alcuni aspetti del modo in cui vengono recuperate le coordinate geografiche.

- `enableHighAccuracy`: proprietà booleana (dunque i soli valori ammissibili sono `true` e `false`). Se impostata a `true` esercita la richiesta di applicare il massimo grado di precisione possibile nel rilevare le coordinate. È da valutare con attenzione se è realmente necessario richiedere acriticamente sempre il massimo livello di accuratezza possibile. Questo ha un duplice costo, sia in termini di tempo (se ne dovrà impiegare di più per ottenere il risultato), sia in termini di consumo energetico sui dispositivi portatili (per esempio smartphones e tablet come iPad o Galaxy Tab), ossia un maggiore dispendio di batterie.

**TABELLA 10.2** Valori ammissibili per la proprietà `code` dell'oggetto `PositionError`

VALORE	CAUSA DELL'ERRORE	DESCRIZIONE
1	<code>PERMISSION_DENIED</code>	Il tracciamento delle coordinate geografiche non ha avuto luogo perché l'utente ha negato il suo consenso.
2	<code>POSITION_UNAVAILABLE</code>	Tracciamento non eseguito a causa di un errore interno (per esempio imputabile a un errore del fornitore dei dati di localizzazione; per inciso, se il browser è Opera, Firefox o, ovviamente, Chrome, il fornitore principale è Google con il suo servizio Google Location Service).
3	<code>TIMEOUT</code>	È stato superato il tempo massimo entro cui il tracciamento avrebbe dovuto avere luogo. Il tempo di <code>TIMEOUT</code> è uno degli elementi oggetto di configurazione, come vedremo più avanti.

- `maximumAge`: nel caso di acquisizione del posizionamento ripetuta nel tempo, si può decidere di non leggere ripetutamente le nuove coordinate se quelle acquisite in precedenza non sono più vecchie di un determinato intervallo di tempo, anche in questo caso espresso in millisecondi.
- `timeout`: se si teme che il tentativo di localizzazione possa impiegare tempi biblici di risposta, si può impostare un intervallo temporale superato il quale la rilevazione del posizionamento del dispositivo non ha più luogo automaticamente (l'utente può sempre essere invitato in via alternativa a "dare una mano" inoltrando il codice di avviamento postale). Questo periodo di tempo è misurato in

millisecondi. Il suo valore predefinito, nel caso in cui non venga impostato esplicitamente, è pari a zero.

## ATTENZIONE

Il tempo che l'utente impiega a dare il suo consenso al tracciamento delle coordinate geografiche non è computato nell'intervallo di tempo misurato con la proprietà `timeout`.

Sfruttando anche le opzioni messe a disposizione di questo terzo argomento potremmo ancora modificare l'invocazione della funzione `getCurrentPosition()` come segue:

### **LISTATO 10.9** Terzo argomento: configurare il processo di tracciamento

```
navigator.geolocation.getCurrentPosition(geolocalizzami, geoErrore, {timeout:20000});
```

In questo caso abbiamo impostato un intervallo di tempo di 20 secondi (il valore è espresso in millisecondi). Usando la notazione descritta, nel Listato 10.11 avremmo potuto attribuire un valore anche per le altre proprietà:

### **LISTATO 10.10** Terzo argomento: configurare il processo di tracciamento, un altro esempio

```
navigator.geolocation.getCurrentPosition(geolocalizzami,  
    geoErrore,  
    {timeout:20000, maximumAge:0, enableHighAccuracy:false}  
) ;
```

Le proprietà `maximumAge` e `enableHighSecurity` sono state impostate esplicitamente ai loro valori predefiniti; ciò è di scarsa utilità, ma qui serve per illustrare come sia possibile utilizzare tutti gli elementi configurabili.

## Tracciamento di un percorso. `watchPosition()` e `clearWatch()`

Per tracciare il posizionamento di un dispositivo nel tempo in modo da evidenziarne il percorso è necessario ricorrere al metodo `watchPosition()`. Se ne osserviamo la firma, a questo punto del capitolo avvertiremo una sensazione di déjà vu.

### **LISTATO 10.11** La firma della funzione `watchPosition()`

```
long watchPosition(in PositionCallback successCallback,  
    in optional PositionErrorCallback errorCallback,  
    in optional PositionOptions options);
```

L'unica vera differenza, oltre ovviamente al nome della funzione, è che essa restituisce un numero. Si tratta di un identificativo univoco che servirà per localizzare i successivi

spostamenti dello stesso dispositivo. Per terminare questa lettura continua delle coordinate geografiche si utilizzerà `clearWatch()`, che accetta come argomento tale numero. Dal momento dell'invocazione di `clearWatch()` ogni successiva acquisizione di coordinate di quel dispositivo deve essere inibita.

In un caso realistico avremmo un codice di questo tipo:

#### **LISTATO 10.12** Seguire un dispositivo e i suoi spostamenti

```
var watchId = watchPosition(successFunction, errorFunction);
[omissis]
// al verificarsi delle condizioni per cui si dispone il
// termine della lettura delle coordinate si avrà:
clearWatch(watchId);
```

La particolarità di questo metodo è che la funzione `successFunction` riceve notifica delle nuove coordinate ogni qualvolta queste subiscano, e qui cito testualmente la specifica, "variazioni significative". Purtroppo non ci è consentito capire che cosa si intenda per variazione significativa; presumo sia da mettere in relazione con il grado di precisione che si riesce a raggiungere.

## Reverse geocoding

Reverse geocoding: ovvero come passare dalle coordinate geografiche a qualcosa di più comprensibile, come il nome di una via o di una località: informazioni decisamente più utili della coppia di valori latitudine e longitudine.

Applichiamo all'esempio visto in precedenza le mappe di Google per dare questo più alto livello di comprensione alle informazioni relative al posizionamento.

Dedicheremo principalmente la nostra attenzione alla funzione `geolocalizzami()`. Avendo deciso di usare allo scopo le mappe di Google (Google Maps) dobbiamo inserire nell'intestazione del documento il seguente riferimento:

#### **LISTATO 10.13** Specifica di esistenza di sensori per la geolocalizzazione

```
<script src="http://maps.google.com/maps/api/js?sensor=false"></script>
```

In questo modo includiamo le API sulle mappe di Google e impostiamo a `false` l'argomento booleano `sensor`. Ciò serve per specificare che non disponiamo di un sensore, quale potrebbe essere, per esempio, il localizzatore satellitare GPS.

Inizialmente usiamo tre variabili per tenere traccia di longitudine, latitudine e precisione

del rilevamento; omettiamo dunque data e ora di rilevamento perché poco utili ai fini di questo esempio.

#### **LISTATO 10.14** Longitudine, latitudine e precisione

```
// leggo coordinate e grado di precisione delle stesse
var lon = position.coords.longitude;
var lat = position.coords.latitude;
var acc = position.coords.accuracy;
acc = parseFloat(acc)/1000 + " km dal punto segnato sulla mappa!"
```

Poiché utilizzeremo più volte questi dati, usiamo delle variabili che ne rendono più comodo il successivo riferimento. La variabile `acc`, come ricorderete, esprime in metri il grado di precisione del posizionamento. Si divide per 1000 questo valore per poterlo riportare in chilometri. Dopo aver inserito questi dati sulla pagina web tramite la proprietà `innerHTML` possiamo usare longitudine e latitudine per identificare un punto sulla mappa.

#### **LISTATO 10.15** Caricamento e configurazione della mappa di Google

```
var latlon = new google.maps.LatLng(lat, lon);
var myOptions = {
  zoom: 15,
  center: latlon,
  mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

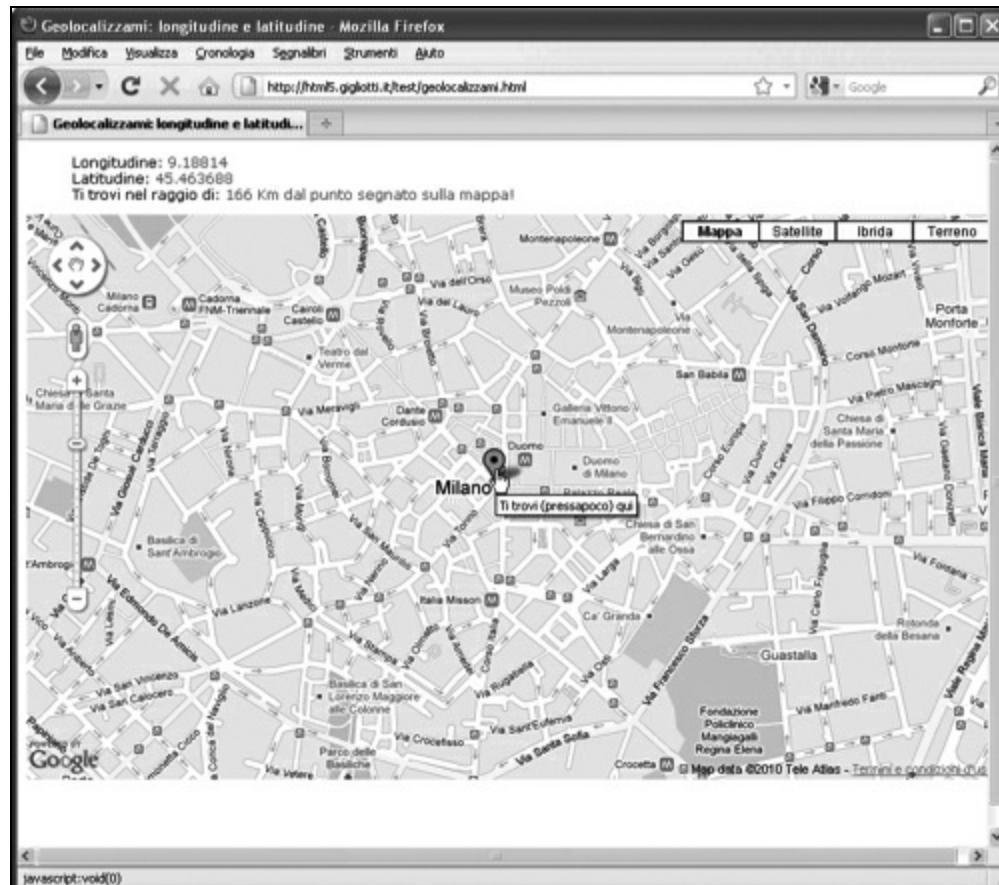
Creando un oggetto `latlon` non facciamo altro che definire un riferimento alla latitudine e longitudine che andremo a visualizzare sulla mappa. Subito dopo si interviene su alcuni elementi di configurazione, in questo caso lo zoom, la posizione del punto di latitudine e longitudine identificati rispetto alla mappa e il tipo di mappa. Esistono altri tre tipi di mappe supportate: `SATELLITE`, `HYBRID` e `TERRAIN`.

#### **LISTATO 10.16** Creazione della mappa e aggiunta del segnaposto

```
var map = new google.maps.Map(document.getElementById("myMap"), myOptions);
var marker = new google.maps.Marker({
  position: latlon,
  map: map,
  title:"Ti trovi (pressapoco) qui"
});
```

Si crea una mappa e la si posiziona all'interno dell'elemento il cui `id` è `myMap`. Si tratta di un elemento `<div>`. Infine si crea un segnaposto (un oggetto di tipo `Marker`) che serve per rendere immediatamente identificabili le coordinate geografiche.

Ed ecco il listato completo che ha prodotto, nel mio caso, il risultato apprezzabile in Figura 10.14.



**FIGURA 10.14** Le coordinate geografiche sono state riportate sulla mappa.

### **LISTATO 10.17** Le coordinate sulla mappa: il listato completo

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Geolocalizzami: longitudine e latitudine</title>
    <script
      src="http://maps.google.com/maps/api/js?sensor=false"></script>
    <script src="modernizr-1.5.min.js"></script>
    <style type="text/css">
      html, body {
        width: 100%;
        height: 100%;
        font: 0.8em verdana, sans-serif;
      }
```

```

#myMap {
    height: 80%;
    width: 100%;
}

span { color: #f00; }
li { list-style-type:none; }

</style>
</head>
<body>
<script>
if (Modernizr.geolocation) {
    navigator.geolocation.getCurrentPosition(geolocalizzami,
    geoErrore, {timeout:20000});
} else {
    alert("geolocalizzazione non supportata dal browser!");
}

function geolocalizzami(position) {
    // leggo coordinate e grado di precisione delle stesse
    var lon = position.coords.longitude;
    var lat = position.coords.latitude;
    var acc = position.coords.accuracy;

    acc = parseFloat(acc)/1000 + " km dal punto segnato sulla mappa!"

    // aggiungo questi dati sulla pagina web
    document.getElementById("lon").innerHTML = lon;
    document.getElementById("lat").innerHTML = lat;
    document.getElementById("acc").innerHTML = acc;
    // caricamento e configurazione della mappa di Google
    var latlon = new google.maps.LatLng(lat, lon);
    var opzioni = {
        zoom: 15,
        center: latlon,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    // creazione della mappa e aggiunta del segnaposto
    var map = new google.maps.Map(document.getElementById("myMap"), opzioni);
    // si posiziona il segnaposto sulla mappa
    var segnaposto = new google.maps.Marker({
        position: latlon,
        map: map,
        title:"Ti trovi (pressapoco) qui"
    });
}

function geoErrore(positionError) {
    alert(positionError.message + " codice[" +
    positionError.code + "]");
}

```

```

</script>
<ul>
<li>Longitudine: <span id="lon">-</span></li>
<li>Latitudine: <span id="lat">-</span></li>
<li>Ti trovi nel raggio di: <span id="acc">-</span></li>
</ul>
<div id="myMap"></div>
</body>
</html>

```

## 166 km: bella precisione!

Siamo onesti: affermare "ti trovi nel raggio di 166 km..." non è proprio un bel risultato. Cerchiamo di capire perché questo accade. Per tracciare il posizionamento del mio PC non sto usando un sensore, come il precisissimo GPS che certamente produrrebbe ben altri risultati. È molto più probabile che le mie coordinate siano state determinate sulla base del mio indirizzo IP che, tuttavia, mi viene assegnato dinamicamente dal provider che mi fornisce il servizio di connettività ADSL

## Disabilitare la geolocalizzazione

Questo paragrafo rappresenta il disperato tentativo di regalarvi un'utile informazione persino nel caso siate terrorizzati dalla geolocalizzazione e temiate la minaccia alla vostra riservatezza da "incalliti geolocalizzatori senza scrupoli". Di seguito, dunque, trovate una breve guida multi-browser su come disabilitare questa funzionalità.

## Firefox

Digitate `about:config` nella barra dell'indirizzo e premete Invio (Figura 10.15).

Fate clic sul pulsante Farò attenzione, prometto (Figura 10.16).

Nella barra del filtro digitate: `geo.enabled` (Figura 10.17).

Un doppio clic sul parametro ne cambia il valore da `true` a `false`, disabilitando la geolocalizzazione.



**FIGURA 10.15** La barra dell'indirizzo di Firefox.



**FIGURA 10.16** Firefox ci avvisa che stiamo per modificare impostazioni avanzate.



**FIGURA 10.17** La modifica del parametro disabilita la geolocalizzazione in Firefox.

## Opera

Fate clic su Preferenze nel menu Strumenti, come illustrato in Figura 10.18.

Nella finestra Preferenze selezionate la scheda Avanzate (Figura 10.19).



**FIGURA 10.18** Il comando Strumenti | Preferenze.



**FIGURA 10.19** La scheda Avanzate è la prima da destra.

Nella sezione posta a destra scegliete la voce Rete e tra le opzioni relative a questa area fate clic sulla casella di controllo Enable geolocation per rimuovere il segno di spunta (Figura 10.20).

#### NOTA

Il fatto che l'etichetta non sia tradotta in italiano è indice di quanto sia ancora recente l'implementazione di questa funzionalità in Opera, disponibile solo dalla versione 10.60. Le immagini sono tratte da una versione 10.62.

## Chrome

Fate clic sull'icona della chiave inglese posta in alto a destra, da cui si accede a un menu utile a personalizzare e controllare il comportamento del browser (Figura 10.21).

Selezzionate la scheda Roba da smanettoni, fate clic su Impostazioni contenuti quindi, nella successiva finestra di dialogo, selezzionate la voce Percorso e inibite la geolocalizzazione mediante l'ultimo radio button.

## Preferenze

Generali Moduli Ricerche Pagine web Avanzate

Schede  
Navigazione  
Notifiche

Contenuti  
Font  
Trasferimenti  
Programmi

Cronologia  
Cookie  
Sicurezza

Rete  
Archivi

Barre strumenti  
Scorciatoie  
Voci

Configurare i server proxy se non si ha una connessione diretta a Internet

Server proxy...

Scegliere cosa fare quando si inserisce una singola parola nel campo indirizzi

Completamento degli indirizzi...

Codifica gli indirizzi web con UTF-8

Invia le informazioni sul server di provenienza

Attiva il rindirizzamento automatico

Enable geolocation

Num. max connessioni server

16 (Predefinito)

Num. max connessioni totali

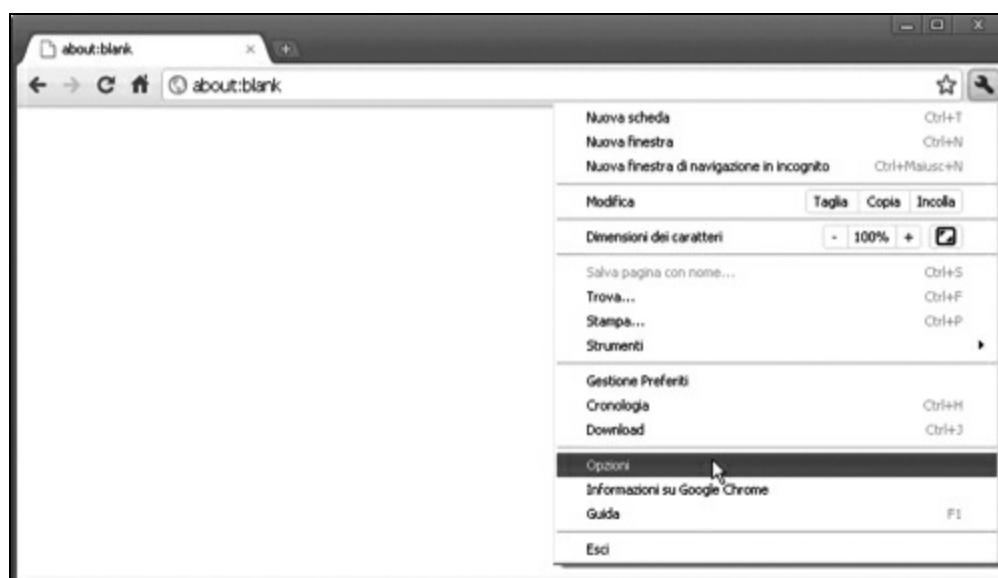
64 (Predefinito)

OK

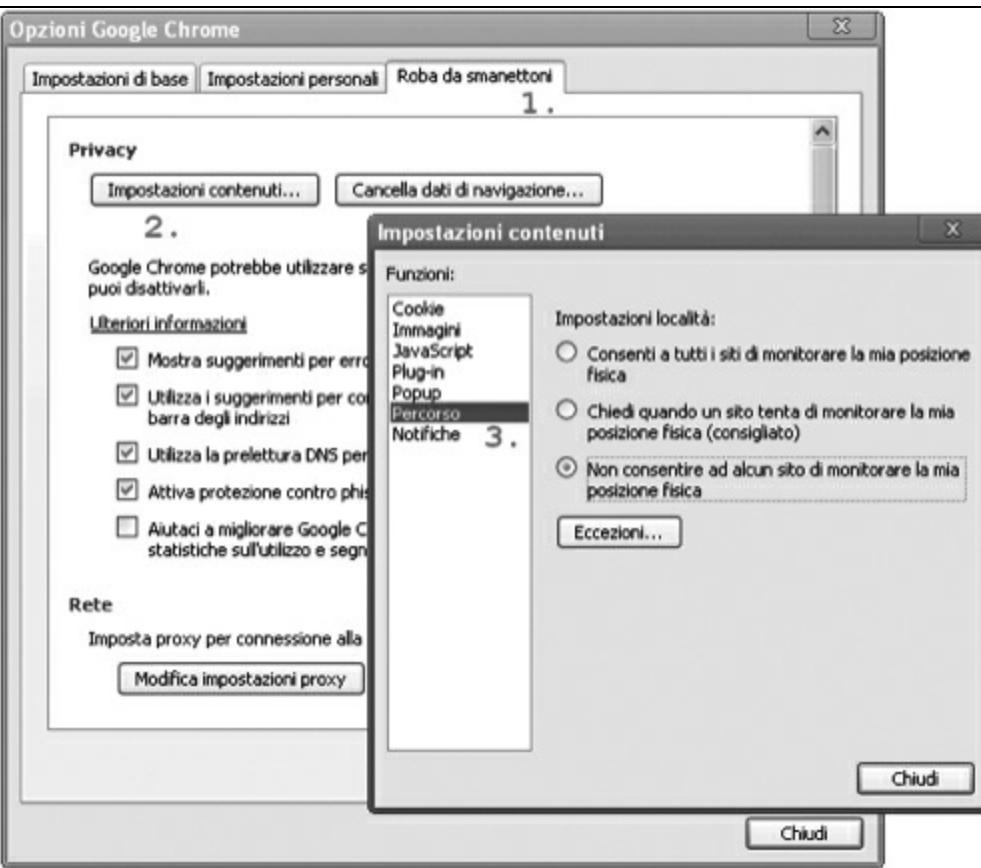
Annulla

Guida

**FIGURA 10.20** Togliendo il segno di spunta dall'apposita casella si disabilita la geolocalizzazione.



**FIGURA 10.21** Il menu con il comando Opzioni.



**FIGURA 10.22** Selezionate l'opzione Non consentire... per inibire la localizzazione.

## Geotiquette

Ottenerne e gestire informazioni così sensibili per la riservatezza di una persona è una faccenda delicata. Inoltre, utilizzare in modo maldestro questi dati porterebbe a risultati che nella migliore delle ipotesi potremmo definire ridicoli. Per questo motivo ho pensato di riassumere di seguito cinque regole che, in omaggio al buon senso e alla tutela dei dati personali, si propongono come guida per chiunque intenda sviluppare un'applicazione che sfrutta la geolocation API descritta in questo capitolo.

- Esplicitare sempre il motivo per cui si intende localizzare il dispositivo dell'utente. Se è evidente che un navigatore satellitare via Web deve necessariamente disporre delle coordinate geografiche di un dato dispositivo per poter erogare il suo servizio, è meno evidente la ragione della richiesta se questa viene dal sito del Corriere della Sera. Beninteso, il Corriere potrebbe chiedere queste informazioni per dare maggiore rilevanza alle notizie geograficamente più vicine all'utente: ciò è legittimo, ma di certo è un servizio accessorio e per questo forse la sua comprensione è meno immediata. Se si riserva un po' di tempo per spiegare quali sono i vantaggi potenzialmente ottenibili aumenta la probabilità che l'utente sia disposto a lasciarsi geolocalizzare... persino quando legge il quotidiano.
- Non impostare `enableHighAccuracy` a `true` (il suo valore predefinito è `false`), a meno che questo non sia strettamente necessario. Sempre rifacendoci all'applicazione web

del navigatore satellitare: questo è un buon esempio in cui è necessario impostare `enableHighAccuracy = true`. Senza la massima precisione possibile (raggiungibile via GPS), difficilmente questo tipo di applicativo potrebbe funzionare senza produrre risultati imbarazzanti. Se invece si lavora a un applicativo che offre contenuti in base a macro-aree geografiche, una stima approssimativa della propria posizione sarà sufficiente nella stragrande maggioranza dei casi. Gli utenti che non si ritengono soddisfatti dovranno essere messi nelle condizioni di correggere il posizionamento rilevato.

- Non conservare le coordinate geografiche. Al contrario, cessate di dispornere appena terminata l'erogazione del servizio. Se intendete storizzare questi dati avvisate gli utenti di questo "dettaglio". In altri termini, non limitatevi alle note in corpo 8 in fondo a una noiosa sfilza di "condizioni del servizio".
- Dismettere le coordinate ottenute se la loro precisione è insoddisfacente ai fini dell'erogazione del servizio. Nell'esempio fatto in questo capitolo, in cui abbiamo riportato le coordinate geografiche su di una mappa, risulta evidente dal valore della proprietà `accuracy` come il posizionamento del mio PC sia stato rilevato con un mediocre grado di precisione. In realtà, benché approssimative, queste informazioni potrebbero rivelarsi più che sufficienti se, per esempio, mi occupo dell'erogazione di un servizio che offre il calendario di tutti i concerti che si terranno in una certa macro-area, per esempio a livello regionale o di aggregato di regioni. Al contrario, questo livello di dettaglio non è adeguato se voglio mostrare le pizzerie disponibili nelle immediate vicinanze delle coordinate rilevate.
- Non chiedere di localizzare l'utente se questi non ne ricava dei reali vantaggi. Solo perché adesso è di moda, si tende a inserire la geolocalizzazione ovunque, indipendentemente dal concreto beneficio che l'utente può trarne. L'utente ne trae una convenienza economica? Risparmia del tempo? Ecco, se la risposta a queste domande è un secco no, probabilmente la geolocalizzazione è inutile.

## Riferimenti alle risorse citate e altri link utili

- La specifica dell'interfaccia di programmazione sulla localizzazione geografica pubblicata dal W3C: <http://www.w3.org/TR/geolocation-API>
- Il componente aggiuntivo Google Gear da installare su Internet Explorer per arricchirlo, tra le altre cose, del supporto a questa interfaccia di programmazione: <http://gears.google.com>
- Adrian Bateman, che ha preso parte per conto di Microsoft al processo di revisione della specifica HTML5, ha chiarito in una lettera di risposta a un utente che in Internet Explorer 9 non è al momento prevista l'implementazione dell'interfaccia di programmazione di geolocalizzazione (la mail di risposta è del 3 settembre 2010): <http://lists.w3.org/Archives/Public/public-geolocation/2010Sep/0004.html>

- L'articolo pubblicato sul Guardian il 23 luglio 2010 dal titolo "Come sono diventato un Foursquare stalker" <http://www.guardian.co.uk/technology/2010/jul/23/foursquare/print>
- È poco utile ai fini dell'apprendimento della geolocalizzazione, comunque modo il titolo Fratello, dove sei? è un omaggio al film dei fratelli Cohen: <http://www.imdb.com/title/tt0190590/>

## Conclusioni

L'interfaccia di programmazione documentata in queste pagine è di una semplicità di utilizzo tale da rendere accessibili le potenzialità di questa tecnologia a una platea molto ampia di autori di pagine web, e tutto ciò deve essere salutato con il massimo entusiasmo. Il supporto di questa API è già ottimo, perché direttamente o indirettamente, attraverso il componente aggiuntivo Gears può dirsi estesa a tutti i principali browser, il che ne incoraggerà ulteriormente la diffusione.

Tuttavia, le implicazioni in termini di tutela della riservatezza dei propri dati che un impiego sempre più pervasivo di queste tecniche comporta dovrebbero spingere i creatori di applicazioni web ad accrescere la consapevolezza dei propri utenti in merito alle informazioni che si troveranno a condividere. Su questo aspetto purtroppo ancora troppo poco è stato fatto.

## Capitolo 11

# CSS: Customizing pages So Strongly (version n. 3)

Il titolo di questo capitolo non poteva essere meno enfatico di quello proposto.

Se HTML5 segna il passaggio verso un linguaggio sempre più votato alla rappresentazione della struttura e della semantica del documento, liberandosi di quel pesante fardello rappresentato dagli elementi stilistici, ciò è stato possibile anche grazie all'affermazione dei fogli di stile. I CSS, il cui acronimo rappresenta le iniziali di Cascading Style Sheets che possiamo tradurre come «fogli di stile a cascata», infatti, rappresentano un naturale complemento del linguaggio di marcatura, fornendo istruzioni su come le pagine web debbano essere rappresentate.

Nonostante i bachi e le interpretazioni fantasiose o tardive di alcuni browser, i fogli di stile hanno saputo conquistare, nel corso degli anni, un ruolo di primo piano tra gli autori di pagine web come strumento di formattazione delle informazioni contenute nei documenti.

È stato un processo lungo, che ha avuto inizio nel 1996 con il rilascio ufficiale della prima versione da parte del W3C. Due anni dopo è stata la volta del CSS2, seguita a otto anni di distanza dal CSS2.1 che fa proprie alcune implementazioni introdotte dai browser più diffusi e rivede in alcuni punti la precedente versione della specifica.

Con il CSS3, il gruppo di lavoro sui fogli di stile ha deciso di optare per un approccio diverso rispetto alle versioni precedenti della specifica: non più una specifica prodotta come un unico blocco monolitico, bensì un approccio modulare. Ciascun modulo tratta un tema diverso inerente i fogli di stile e, di conseguenza, ciascun tema è sviluppato con tempistiche diverse. Per questo motivo non ha molto senso parlare genericamente di supporto al CSS3. Come già fatto per la sezione del manuale relativo a HTML, si soprassiede qui sulle basi del linguaggio, per le quali troverete un'ottima guida in un altro testo della stessa collana: CSS, Guida completa di Gianluca Troiani. Qui ci limitiamo a prendere in esame alcune delle novità dell'ultima versione del linguaggio.

- Strumenti tipografici: maggiore controllo sul testo; gestione avanzata della disposizione di testo, immagini ed elementi multimediali su più colonne; opacità degli elementi, bordi con angoli smussati.
- Effetti grafici: transizioni, trasformazioni e animazioni.

Riguardo al supporto, tra i principali browser assistiamo a una risposta positiva da parte di Firefox, Chrome, Safari e Opera, mentre le difficoltà maggiori si riscontrano nel mondo Microsoft, dove le versioni 6, 7 e 8 si distinguono per una implementazione carente, negli scenari migliori. A questa situazione pone rimedio Internet Explorer 9 che sin dal rilascio della sua prima beta, il 15 settembre del 2010, annunciato con grande entusiasmo dalla casa di Redmond, si è fatta vanto di un alto grado di supporto per questa versione della specifica. Ciò rappresenta decisamente una spinta significativa alla diffusione del CSS3, tuttavia occorre ricordare, con sano realismo, che il ciclo di vita di ciascuna versione del browser Internet Explorer è infinitamente più lungo di quello di ogni altro browser concorrente; questo non deve però farci giungere alla conclusione che si debba rinunciare a queste novità. Sarà sufficiente dedicarsi all'applicazione di strategie di supporto alternativo che possano premiare gli utenti di browser di ultima generazione e, in generale, quelli più ligi agli standard, senza dover trascurare tutti gli altri. Questi ultimi, pur non potendo godere di un'esperienza di navigazione evoluta, non perderanno comunque accesso a quanto ritengono più prezioso: le informazioni che state pubblicando.

**TABELLA 11.1** Supporto dei selettori avanzati CSS3

ARGOMENTO	BROWSER
Selettori avanzati CSS3	Chrome 5+, Firefox 3.5+, IE9, Opera 10.1, Safari 3.2

## Nuovi selettori e pseudo-classi

Uno dei vantaggi dei nuovi selettori di attributo e delle nuove pseudo-classi è di riuscire a intervenire in modo puntuale e discreto. I due aggettivi non sono scelti a caso. Le nuove pseudo-classi possono applicarsi in modo puntuale perché anche di fronte di una pagina web arbitrariamente complessa riescono a intervenire su di un elemento ben definito, per esempio un elemento `<tr>` che rappresenti la quart'ultima riga di una tabella. I nuovi selettori di attributo possono identificare con estrema sintesi e precisione elementi che condividono anche solo una parte del valore assegnato a un loro attributo. È proprio sfruttando uno dei nuovi selettori di attributo che, per esempio, possiamo applicare uno stile diverso a un link in base al tipo di dominio (`.it, .org, .com, .edu`) cui è offerto il collegamento.

La regola CSS, inoltre, trova applicazione in modo discreto, nel senso di non invasivo, perché può essere applicata senza dover intervenire sulla struttura HTML con un attributo `class` o `id`. Queste caratteristiche rendono ancora più evidente la desiderabilità dei CSS in un'ottica di pulizia del codice e conseguentemente in termini di contenimento dei costi di manutenzione del codice stesso.

# Selettori di attributo

I selettori di attributo sono già da tempo parte della specifica. Essi permettono di impostare una regola CSS che si applichi a un dato elemento HTML in base alla presenza di un particolare attributo o al valore che questi può assumere. La Tabella 11.2 riassume i selettori già disponibili, mentre nei prossimi paragrafi saranno analizzati i nuovi selettori di attributo.

**TABELLA 11.2** Selettori di attributo

SELETTORE DI ATTRIBUTO	DESCRIZIONE	ESEMPIO
elem[attr]	La regola si applica all'elemento di tipo <code>elem</code> che impiega l'attributo <code>attr</code> .	<code>p[id]</code> Si applica a: <code>&lt;p id="intro"&gt;Lorem ipsum&lt;/p&gt;</code> Non si applica a: <code>&lt;p&gt;...&lt;/p&gt;&lt;p itemscope&gt;...&lt;/p&gt;</code>
elem[attr="val"]	La regola si applica all'elemento di tipo <code>elem</code> al cui attributo <code>attr</code> sia stato assegnato il valore <code>"val"</code> .	<code>a[media="print"]</code> Si applica a: <code>&lt;a href="aboutme.html" media="print"&gt;...&lt;/a&gt;</code> Non si applica a: <code>&lt;a href="aboutme.html" media="handheld"&gt;...&lt;/a&gt;</code>
elem[attr~="val"]	La regola si applica all'elemento di tipo <code>elem</code> al cui attributo <code>attr</code> sia stato assegnato un valore composto da una lista di parole separate da spazi e una di esse sia esattamente uguale a <code>"val"</code> .	<code>img[title~="grazie"]</code> Si applica a: <code>&lt;img src="min.jpg" title="tre grazie minorchine"&gt;</code> Non si applica a: <code>&lt;a title="3grazie minorchine"&gt;...&lt;/a&gt;</code>
elem[attr = "val"]	La regola si applica all'elemento di tipo <code>elem</code> al cui attributo <code>attr</code> sia stato assegnato un valore composto da una lista di parole separate da un trattino e la prima di questa lista corrisponda esattamente a <code>"val"</code> .	<code>time[datetime ="2011"]</code> Si applica a: <code>&lt;time datetime="2011-09-01"&gt;</code> Una giornata vissuta con stile</time> Non si applica a: <code>&lt;time datetime="2010-05-02"&gt;2 maggio 2010&lt;/time&gt;</code>

Questo selettori è in grado di identificare tutti gli elementi di tipo `elem` per cui è definito l'attributo `attr` e il cui valore inizi per `val`. Ciò significa che la regola espressa nel Listato 11.1 si applicherà solo al primo dei due link riportati nel Listato 11.2.

## **LISTATO 11.1** elem[attr^="val"]. Un esempio

```
a[href^="mailto:"] {text-decoration:none; color: green; font-weight: bold;}
```

## **LISTATO 11.2** elem[attr^="val"]. Campo di applicazione

```
<footer>
  Per contattarmi puoi scrivermi una mail indirizzandola a
  <a href="mailto:scrivimi@example.org">scrivimi@example.org</a>
  oppure puoi compilare il form disponibile presso la pagina dei
  <a href="http://www.example.org/contattami.php">contatti</a>
</footer>
```

In questo caso la regola CSS sarà applicata solo al primo dei due collegamenti ipertestuali, perché solo il primo dei due identifica un “URL mailto”, ossia un collegamento ipertestuale a un indirizzo di posta elettronica.

elem[attr\*="val"]

La regola così definita si applica all’elemento di tipo `elem` il cui attributo `attr` assume un valore che contiene la parola “`val`”.

## **LISTATO 11.3** elem[attr\*="val"]. Un esempio

```
time[datetime*="-09-"] {color: green; font-weight: bold;}
```

## **LISTATO 11.4** elem[attr\*="val"]. Campo di applicazione

```
<p> Era il <time datetime="2010-09-01">primo settembre 2010</time> ... </p>
<p>Un mese prima, il <time datetime="2011-08-01">primo agosto 2010</time>... </p>
<p>Due mesi prima, il <time datetime="2011-07-01">primo luglio 2010</time> ...</p>
```

La regola viene applicata a tutti gli elementi `<time>` che presentano un attributo `datetime` per cui sia impostata una data che si riferisce al mese di settembre.

elem[attr\$="val"]

Una regola così definita si applica a tutti gli elementi di tipo `elem` il cui attributo `attr` abbia un valore che termina per `val`. Un interessante campo di applicazione potrebbe essere quello dei collegamenti ipertestuali per cui la regola trova applicazione in base al dominio (`it, com, org, edu` ecc.).

## **LSTATO 11.5** elem[attr\$="val"]. Un esempio

```
a[href$=".edu"] {color: red; font-weight: bold;}
```

## **LSTATO 11.6** elem[attr\$="val"]. Campo di applicazione

```
<p><a href="http://www.example.edu" title="edu link">conoscenza</a></p>
<p><a href="http://www.example.com" title="com link">denaro</a></p>
<p><a href="http://www.example.org" title="org link">solidarietà</a></p>
```

## **ATTENZIONE**

La facoltà di intervenire in modo così capillare nell’attribuzione di uno stile ha un prezzo che si paga in termini di performance. Calcolare quali e quanti elementi siano interessati da una regola di stile così definita infatti è oneroso. L’autore Steve Souders, nel suo manuale *Even Faster Web Sites*, edito da O’Reilly, dedica il paragrafo “Writing Efficient CSS Selectors” all’argomento. Souders stila una serie di otto regole. La prima di esse recita “Evita i selettori universali”, tra i quali egli annovera i selettori di attributo che abbiamo qui introdotto. Non intendo qui raffreddare il legittimo entusiasmo per questi nuovi selettori di attributi; vale però la pena ricordarsi di quale impatto possano avere e, conseguentemente, riservarne l’utilizzo ai casi davvero utili.

## Pseudo-classi di struttura

Le pseudo-classi di struttura permettono di attribuire uno stile a un determinato elemento in base alla posizione che questi occupa nella struttura del documento HTML.

### elem:root

Contrassegna l’elemento radice di un documento. Per i documenti HTML, questo elemento corrisponde sempre al tag `<html>`. Ciò accade persino quando l’elemento è omesso. Infatti, in tal caso il browser ipotizza che sia comunque stato definito.

### elem:empty

La pseudo-classe `elem:empty` si applica all’elemento `<elem>` solo se quest’ultimo non presenta nodi “figli” al suo interno, ossia se in esso non è innestato né del testo né altri elementi HTML.

## **LSTATO 11.7** elem:empty. Un esempio

```
p:empty {margin: 3.5em;}
```

## **LSTATO 11.8** elem:empty. Campo di applicazione

```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
```

```
dolore magna aliquam erat volutpat.</p>
```

```
<p></p>
```

```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
```

La regola si applica solo al secondo dei tre paragrafi, poiché solo quest'ultimo è vuoto: non contiene né testo né altri elementi. In altri termini, i casi presentati nel Listato 11.9 sono casi di esclusione della regola descritta nel Listato 11.7.

#### **LISTATO 11.9** elem:empty. Casi di esclusione

```
<p>Lorem ipsum dolor sit amet</p>
```

```
<p><em>Lorem ipsum</em></p>
```

```
<p><em>Lorem ipsum</em> dolor sit amet</p>
```

#### elem:only-child

La regola CSS si applica ai soli casi in cui l'elemento al quale è associata la pseudo-classe è l'unico figlio di un elemento HTML padre.

#### **LISTATO 11.10** elem:only-child. Un esempio

```
span:only-child {font-size: 1.5em;}
```

I Listati 11.11 e 11.12 illustrano rispettivamente un campo di applicazione e un caso di esclusione.

#### **LISTATO 11.11** elem:only-child. Campo di applicazione

```
<p>Lorem ipsum dolor <span class="esempio">sit amet</span> ...</p>
```

#### **LISTATO 11.12** elem:only-child. Caso di esclusione

```
<p>Lorem ipsum <strong>dolor</strong> <span class="esempio">sit amet</span> ...</p>
```

#### elem:only-of-type

Si applica a quell'elemento che risulta essere l'unico elemento, per tipo, annidato all'interno dell'elemento padre.

#### **LISTATO 11.13** elem:only-of-type. Un esempio

```
a:only-of-type {text-decoration: none;}
```

Una regola così definita si applica al Listato 11.14 perché il paragrafo, pur contenendo al suo interno altri elementi, presenta un solo collegamento ipertestuale, dunque un solo marcatore di tipo `<a>`. Nel Listato 11.15 questa condizione non è rispettata: esistono due collegamenti. La regola di stile che si basa sulla pseudo-classe `:only-of-type` non è applicabile.

#### **LISTATO 11.14** elem:only-of-type. Campo di applicazione

```
<p>Lorem ipsum dolor sit amet, <a href="http://example.org">consectetuer adipiscing elit</a>, sed diam <mark>nonummy</mark> nibh euismod tincidunt ut laoreet dolore magna aliquam <em>erat volutpat</em>. </p>
```

#### **LISTATO 11.15** elem:only-of-type. Caso di esclusione

```
<p>Lorem ipsum dolor sit amet, <a href="http://example.org">consectetuer adipiscing elit</a>, sed diam <mark>nonummy</mark> nibh euismod tincidunt ut laoreet dolore <a href="http://example.org/yet-another-doc.html">magna aliquam <em>erat volutpat</em>. </a></p>
```

### elem:nth-child(n)

Si applica all'ennesimo figlio dell'elemento padre. Ipotizzando di avere l'elenco delle canzoni dell'album *Feels like home* di Norah Jones in una lista numerata e volendo evidenziare la mia preferita, l'ottava canzone della lista, scrivo il Listato 11.16 che produce il risultato apprezzabile in Figura 11.1.

#### **LISTATO 11.16** elem:nth-child. Esempio #1

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>nth-child</title>
    <style>
      body {font-family: sans-serif;}
      ol {width: 30%;}
      li:nth-child(8) {color: red;}
    </style>
  </head>
  <body>
    <article>"Toes" è la mia canzone preferita dell'album "feels like home"
    <ol>
      <li>Sunrise</li>
      <li>What Am I to You?</li>
      <li>Those Sweet Words</li>
```

```

<li>Carnival Town</li>
<li>In the Morning</li>
<li>Be Here to Love Me</li>
<li>Creepin' In</li>
<li>Toes</li>
<li>Humble Me</li>
<li>Above Ground</li>
<li>The Long Way Home</li>
<li>The Prettiest Thing</li>
<li>Don't Miss You At All</li>
</ol>
</article>
</body>
</html>

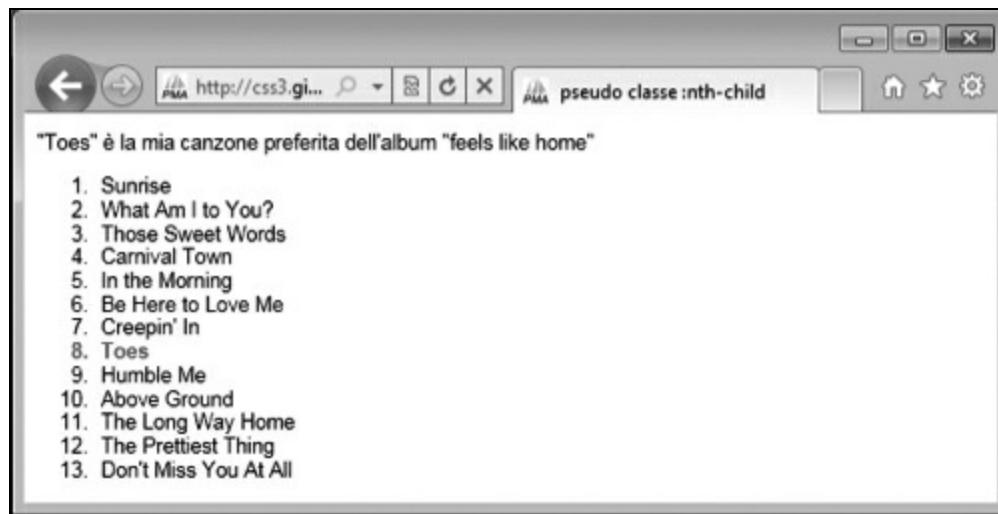
```

Tuttavia, la pseudo-classe `:nth-child` è ancora più versatile perché non agisce solo sull'ennesimo elemento (nel Listato 11.16  $n$  è uguale a 8) ma, in alternativa, può essere applicata a tutti gli elementi che soddisfino una data espressione del tipo  $an+b$ .

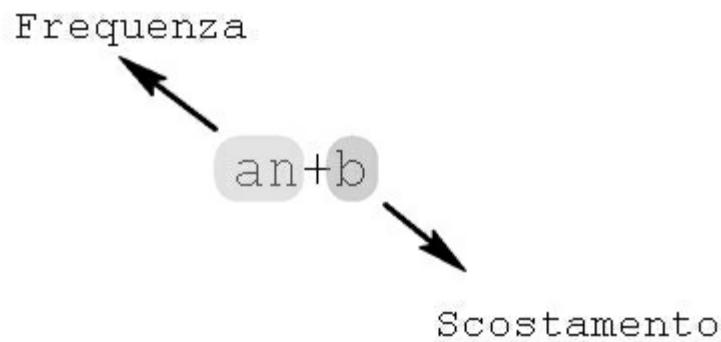
Ipotizziamo di aggiungere anche la regola:

**LISTATO 11.17** `elem:nth-child(an+b)`. Un esempio

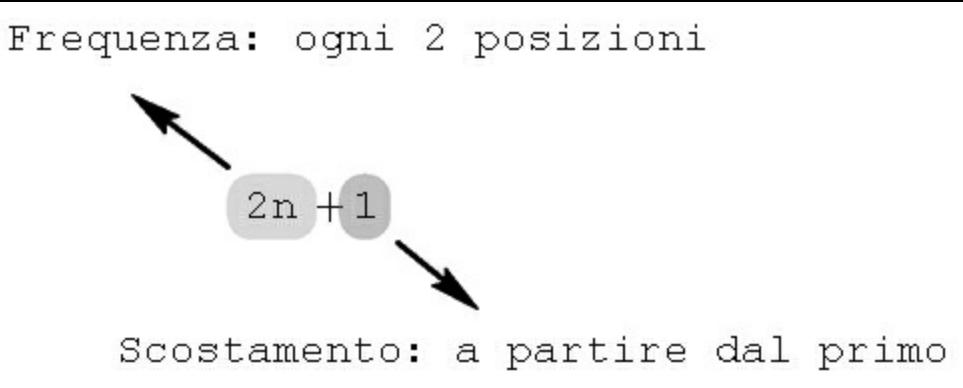
```
li:nth-child(2n+1) {background: #ccc;}
```



**FIGURA 11.1** La regola agisce in modo mirato solo sull'ottavo elemento della lista.



**FIGURA 11.2** Frequenza e scostamento sono determinati attraverso la formula  $an+b$ .

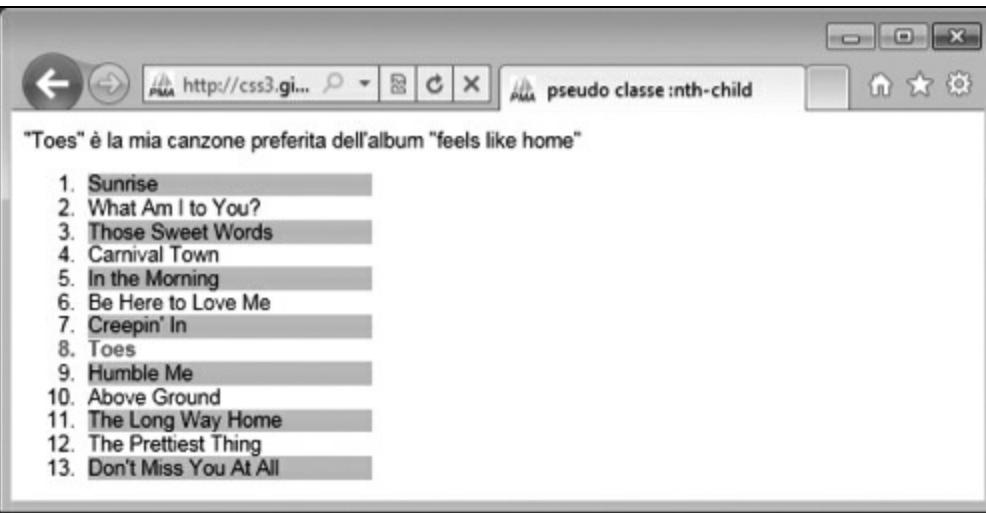


**FIGURA 11.3** Come interpretare l'espressione  $2n+1$ .

L'espressione  $2n+1$  può leggersi come "applica la regola ogni 2 elementi partendo dal primo" (Figura 11.3). La Tabella 11.3 illustra come, in concreto, viene calcolata la posizione di ciascun elemento cui applicare la regola CSS nel caso si utilizzi la formula  $2n+1$ .

**TABELLA 11.3** Calcolo della posizione dei primi tre elementi cui applicare lo stile

VALORE DI N	<b>AN+B = POSIZIONE DELL'ELEMENTO CUI APPLICARE LO STILE</b>
n = 0	$2 * 0 + 1 = 1$
n = 2	$2 * 1 + 1 = 3$
n = 3	$2 * 2 + 1 = 5$



**FIGURA 11.4** Lettura facilitata applicata ad un elenco attraverso la regola `nth-child(an+b)`.

Come si sarà già intuito, questo equivale ad attribuire lo stile a ciascuna riga dispari dell'elenco (Figura 11.4).

Nella formula `an+b`, le due componenti, ossia `an` e `b`, sono impostabili anche separatamente. Nel primo esempio, `li:nth-child(8) {font-weight: bold; color: red;}`, (Listato 11.16) non abbiamo fatto altro che impostare solo lo scostamento, ossia il valore `b` dell'espressione omettendo la frequenza (`an`). Al contrario, possiamo decidere di impostare solo la parte `an`.

Per esempio, avremmo potuto scrivere il Listato 11.17 come:

**LISTATO 11.18** `elem:nth-child(an)`. Un esempio

```
li:nth-child(2n) {background: #ccc;}
```

Così facendo avremmo ottenuto un'applicazione della regola ogni due elementi senza alcuno scostamento, quindi il primo ad avere lo sfondo grigio sarebbe stato l'elemento numero 2 della lista, il secondo sarebbe stato l'elemento numero 4 della lista e così via.

Questi casi particolari, `2n` e `2n+1`, permettono di applicare una regola CSS a elementi alterni: ciò che cambia è solo il numero, pari nel primo caso, dispari nel secondo. Poiché questi sono i casi più frequenti di applicazione, è stata introdotta una sintassi più amichevole (Tabella 11.4).

**TABELLA 11.4** Sintassi alternativa per definire una lettura facilitata

SCRIVERE...	... EQUIVALE A SCRIVERE
<code>elem:nth-child(n2);</code>	<code>elem:nth-child(even);</code>
<code>elem:nth-child(n2+1);</code>	<code>elem:nth-child(odd);</code>

“`even`” e “`odd`” sono i termini inglesi per pari e dispari. Coloro i quali tra voi amano poco la matematica si chiederanno perché li ho torturati inutilmente con il dettaglio dell’espressione `an+b` quando è possibile saltarla a piè pari e impostare la regola in modo più umano. Ebbene, conoscere il funzionamento dell’espressione `an+b` vi offre un grado di libertà maggiore perché vi consente di applicare gli stili adottando frequenze diverse dall’“uno sì e uno no” alle quali rimarreste inchiodati applicando la sintassi facilitata.

## NOTA

Potete esplorare ulteriormente le possibilità offerte da questa espressione scegliendo valori negativi tanto per la frequenza (la parte `an` dell’espressione), quanto per lo scostamento (la parte `b` dell’espressione). Inoltre, ricordate che la lista degli elementi non rappresenta, come spesso accade in programmazione, un indice a base zero: l’indice del primo elemento della lista è 1.

### elem:nth-last-child(n)

Questa pseudo-classe funziona allo stesso modo di `nth-child(n)`, con un’unica differenza: gli elementi figli sono contati a partire dall’ultima occorrenza. Rifacendoci al Listato 11.16, se al posto di scrivere `li:nth-child(8) {color: red;}` avessi scritto:

#### **LISTATO 11.19** elem:nth-last-child(an). Un esempio

```
li:nth-last-child(8) {color: red;}
```

avrei applicato lo stile al sesto elemento `<li>Beh here to love me</li>` (ossia  $13 - 8$ ), perché avrei determinato l’ottavo elemento a partire dal tredicesimo e spostandomi di otto posizioni via via verso l’alto.

### elem:nth-of-type(n)

Anche questa pseudo-classe può considerarsi una variante di `nth-child(n)`; infatti opera allo stesso modo ma se ne differenzia perché prende in esame solo gli elementi del medesimo tipo. Può essere applicata a una serie di immagini distribuite in sequenza in un testo in modo da disporle alternativamente a destra e a sinistra del testo.

Il Listato 11.20 propone un esempio di implementazione e la Figura 11.5 ne illustra il risultato.

#### **LISTATO 11.20** elem:nth-of-type(n). Un esempio

```
<!DOCTYPE html>
<html>
  <head>
```

```
<meta charset="utf-8">
<title>pseudo classe :nth-child</title>
<style>
body {font-family: sans-serif; font-size: 1.2em;}
img {margin: 0em 0.4em 0.5em}
img:nth-of-type(2n+1) { float: right; }
img:nth-of-type(2n) { float: left; }
</style>
</head>
<body>
<article>
<header>
<h1><a href="valencia-in-un-giorno.html">Valencia in un giorno</a></h1>
<p>Scritto da Gabriele Gigliotti</p>
<p>Pubblicato il <time pubdate datetime="2009-11-07">07-11-2010</time>. </p>
</header>

<p>Abbiamo lasciato Bergamo una fredda mattina di tardo ottobre. Erano le 7.45. Saremmo ritornati tredici ore più tardi. Due ore dopo ci stavamo godendo i venti gradi con cui Valencia ci ha accolto sin da subito. Il cambio di temperatura è stato così eclatante e raggiunto in così poco tempo che abbiamo avuto la sensazione di fare un viaggio nel tempo più che nello spazio.<br>Ci sembrava di essere tornati in primavera!</p>



<p>Routard alla mano e guidati da un infallibile senso dell'orientamento <em> (non il mio, decisamente non il mio)</em> abbiamo raggiunto presto il centro della città dove abbiamo visitato il Mercado Central (non prima di aver assaggiato una buona ensaimada). Lì abbiamo scattato le prime foto e abbiamo comprato frutta e acqua per il resto della giornata. Lasciato il mercato abbiamo fatto tappa alla cattedrale</p>



<p>Dalla sua torre "La Miguelete" si apprezza una stupenda vista della città. Nel frattempo era quasi diventata ora di pranzo, lo stomaco brontolava e il Riu era abbastanza vicino. Abbiamo aspettato parecchio prima di assaggiare una paella memorabile, ne è valsa la pena :-(</p>



<p>Abbiamo trascorso il pomeriggio facendo una passeggiata lungo quello che un tempo era il letto del fiume Turia e che è stato convertito in un parco che si estende per chilometri (fossimo stati in Italia avrebbe avuto la meglio il piano alternativo di trasformare il letto del torrente in un'autostrada). Lungo questo torrente si trova la "Ciutat de les Arts i les Ciències" un complesso architettonico concepito da Calatrava che è diventato uno dei simboli della città</p>
<footer>
<p><a href="valencia-in-un-giorno.html" title="L'intero articolo è composto da 1000 parole e 10 immagini">[Leggi la versione integrale]</a><br>
<em>Per segnalare errori di stampa o suggerimenti per prossimi articoli invia una mail all'indirizzo</em>
<a href="mailto:gabriele.gigliotti@gmail.com">gabriele.gigliotti@gmail.com</a></p>
```

```
</footer>  
</article>  
</body>  
</html>
```

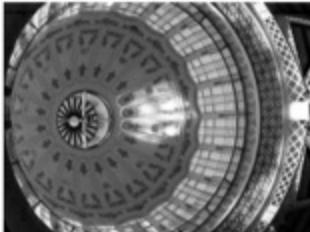
The screenshot shows a web browser window with the URL <http://css3.gigliotti.it/test/pseudo-nth-child>. The page title is "Valencia in un giorno". Below the title, it says "Scritto da Gabriele Gigliotti" and "Pubblicato il 07-11-2010". The main text describes a trip from Bergamo to Valencia. It includes several images: a circular view of a cathedral interior, a collage of Valencia cityscapes, a view from a tower, a paella dish, and the modern architecture of the Turia River. The text is styled using CSS pseudo-classes to position images relative to the text blocks.

**Valencia in un giorno**

Scritto da Gabriele Gigliotti

Pubblicato il 07-11-2010.

Abbiamo lasciato Bergamo una fredda mattina di tardo ottobre. Erano le 7.45. Saremmo ritornati tredici ore più tardi. Due ore dopo ci stavamo godendo i venti gradi con cui Valencia ci ha accolto sin da subito. Il cambio di temperatura è stato così eclatante e raggiunto in così poco tempo che abbiamo avuto la sensazione di fare un viaggio nel tempo più che nello spazio. Ci sembrava di essere tornati in primavera!



Routard alla mano e guidati da un infallibile senso dell'orientamento (*non il mio, decisamente non il mio*) abbiamo raggiunto presto il centro della città dove abbiamo visitato il Mercado Central (non prima di aver assaggiato una buona ensaimada). Li abbiamo scattato le prime foto e abbiamo comprato frutta e acqua per il resto della giornata. Lasciato il mercato abbiamo fatto tappa alla cattedrale



Dalla sua torre "La Miguelete" si apprezza una stupenda vista della città. Nel frattempo era quasi diventata ora di pranzo, lo stomaco brontolava e il Riu era abbastanza vicino. Abbiamo aspettato parecchio prima di assaggiare una paella memorabile, ne è valsa la pena :-)



Abbiamo trascorso il pomeriggio facendo una passeggiata lungo quello che un tempo era il letto del fiume Turia e che è stato convertito in un parco che si estende per chilometri (fossimo stati in Italia avrebbe avuto la meglio il piano alternativo di trasformare il letto del torrente in un'autostrada). Lungo questo torrente si trova la "Ciutat de les Arts i les Ciències" un complesso architettonico concepito da Calatrava che è diventato uno dei simboli della città



[\[Leggi la versione integrale\]](#)

Per segnalare errori di stampa o suggerimenti per prossimi articoli invia una mail all'indirizzo [gabriele.gigliotti@gmail.com](mailto:gabriele.gigliotti@gmail.com)

**FIGURA 11.5** Le immagini sono presentate alternativamente a destra e a sinistra rispetto al testo.

## elem:nth-last-of-type(n)

La sola differenza tra le pseudo-classi `nth-of-type` e `nth-last-of-type` è che quest'ultima impiega un conteggio che parte dall'ultimo elemento verso il primo.

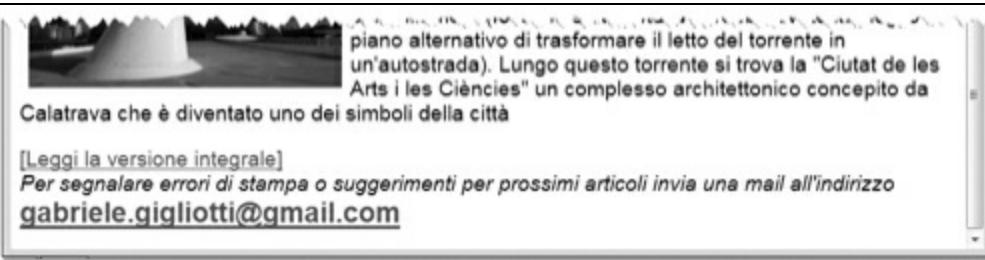
## elem:last-child

La pseudo-classe applica la regola all'ultimo elemento figlio innestato in un elemento contenitore, o elemento padre. Possiamo modificare il modo in cui verrà visualizzato

l'ultimo collegamento ipertestuale in Figura 11.5 aggiungendo al Listato 11.18 la seguente regola e ottenendo il risultato illustrato in Figura 11.6.

### **LISTATO 11.21** elem:last-child. Un esempio

```
a:last-child {font-weight: bold; font-size: 1.3em;}
```



**FIGURA 11.6** L'ultimo collegamento ipertestuale è l'obiettivo della regola definita attraverso la pseudo-classe last-child.

### elem:first-of-type

Equivale a scrivere `nth-of-type(1)` poiché si riferisce al primo elemento di tipo `elem` contenuto in un elemento padre.

### elem:last-of-type

Equivale a scrivere `nth-last-of-type(1)` poiché si riferisce all'ultimo elemento di tipo `elem` contenuto in un elemento padre. Potremmo aggiungere questa pseudo-classe al Listato 11.20 per aumentare lo spazio tra il testo dell'ultimo paragrafo di ogni sezione da cui è composto l'articolo (intestazione, corpo e piè di pagina) e il primo elemento della sezione successiva.

### **LISTATO 11.22** elem:last-of-type. Un esempio

```
p:last-of-type {padding-bottom: 2em;}
```

## Impostare una posizione: le tre opzioni

Abbiamo visto come tutte le pseudo-classi di struttura che accettano una posizione come argomento, ossia `nth-child(n)`, `nth-last-child(n)`, `nth-of-type(n)`, `nth-last-of-type(n)`, possano esprimere questo valore in tre diversi modi:

- attraverso un numero, per esempio: `li:nth-child(8) {color: red;}`
- attraverso una espressione, per esempio: `li:nth-of-type(2n+1) {color: #ccc;}`
- attraverso una parola chiave, per esempio: `li:nth-last-child(even) {color: #ccc;}`

# Altre pseudo-classi

Oltre alle pseudo-classi strutturali sopra elencate, ne sono state introdotte altre che passiamo in rassegna in questo paragrafo.

## elem:target

Si tratta di una pseudo-classe dinamica; appartiene dunque alla stessa categoria delle pseudo classi `:link`, `:visited`, `:active`, `:hover`, `:focus`. Agisce sull'elemento "ancora" che rappresenta la destinazione del link. Il Listato 11.23 e la Figura 11.7 aiutano a capire il suo funzionamento.

### **LISTATO 11.23** elem:target. Un esempio

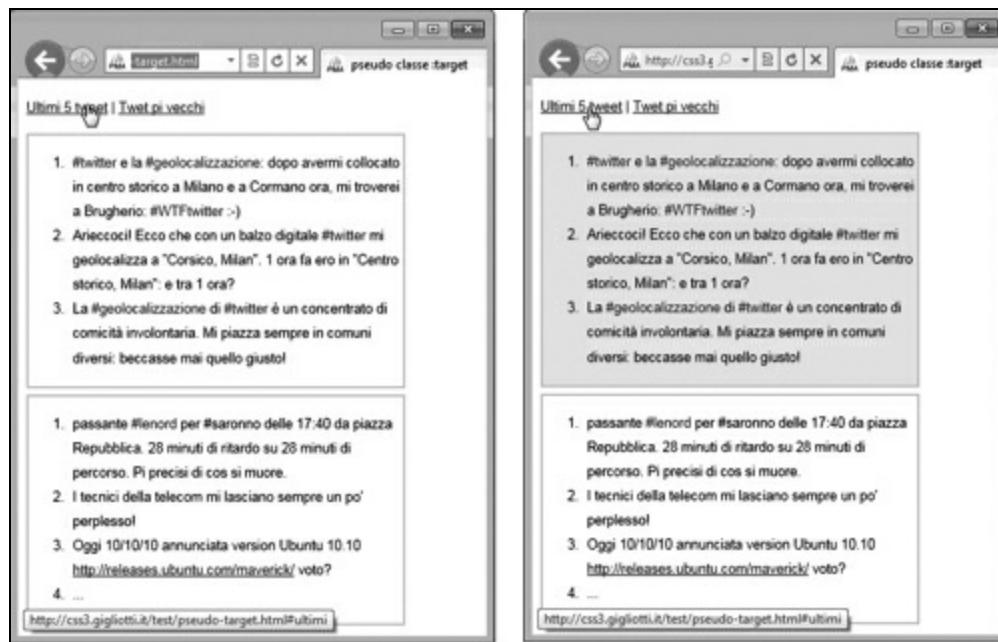
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>pseudo classe :target</title>
    <style>
      body {font: 0.8em/1.8em arial, sans-serif}
      div {width: 350px;
            border: 2px solid #ccc;
            margin: 6px 0px; padding: 2px;}
      em {font-style: normal;
           color: rgb(153,0,0);}
      div:target {background: #efefef;}
      a {color: rgb(153,0,0);}
    </style>
  </head>
  <body>
    <p><a href="#ultimi">Ultimi 5 tweet</a> | <a href="#meno-recenti">Twet più vecchi</a></p>
<div id="ultimi">
  <ol>
    <li><em>#twitter</em> e la <em>#geolocalizzazione</em>: dopo avermi collocato in centro storico a Milano e a Cormano ora, mi troverei a Brugherio: <em>#WTFtwitter</em> :-)</li>
    <li>Arieccoci! Ecco che con un balzo digitale <em>#twitter</em> mi geolocalizza a "Corsico, Milan". 1 ora fa ero in "Centro storico, Milan" e tra 1 ora?</li>
    <li>La <em>#geolocalizzazione</em> di <em>#twitter</em> è un concentrato di comicità involontaria. Mi piazza sempre in comuni diversi: beccasse mai quello giusto!</li>
  </ol>
</div>
<div id="meno-recenti">
  <ol>
    <li>passante <em>#lenord</em> per #saronno delle 17:40 da piazza Repubblica. 28 minuti di ritardo su 28 minuti di percorso. Più precisi di così si muore. </li>
```

```

<li>I tecnici della telecom mi lasciano sempre un po' perplesso!</li>
<li>Oggi 10/10/10 annunciata version Ubuntu 10.10 <a href="http://releases.ubuntu.com/maverick/" title="Ubuntu
10.10 nome in codice Maverick Meerkat"> http://releases.ubuntu.com/maverick/</a> voto?</li>
<li>...</li>
</ol>
</div>
</body>
</html>

```

Nella Figura 11.7 a sinistra la pagina è stata catturata un attimo prima che il link fosse attivato. A destra si osserva che cosa accade subito dopo questo istante. La regola CSS definita mediante la pseudo-classe `:target` diventa effettiva e, nel caso specifico, cambia lo sfondo dell'elemento `<div>` che rappresenta la destinazione del collegamento ipertestuale.



**FIGURA 11.7** La pseudo-classe dinamica `:target` in azione su Internet Explorer 9 beta.

## elem:enabled, elem:disabled, elem:checked

Queste tre pseudo-classi trovano il loro naturale campo di applicazione tra gli elementi dei form. È in quest'ambito, infatti, che si concentrano quegli elementi che possono essere in uno dei tre stati: abilitato, disabilitato o contrassegnato.

### **LISTATO 11.24** elem:enabled, elem:disabled e elem:checked. Un esempio

```

input[type="text"]:enabled {border: 2px solid #900;}
input[type="search"]:disabled {border: 2px solid #ccc;}
:checked {margin: 30px}

```

La prima regola, da applicarsi ai campi di testo che siano abilitati, definisce un bordo di 2 pixel composto da una linea continua di una sfumatura amaranto.

La seconda regola, da applicarsi a un campo di ricerca che sia disabilitato, definisce un bordo di 2 pixel composto da una linea continua di colore grigio scuro.

La terza, improbabile, regola CSS stabilisce che qualunque elemento possa essere contrassegnato, e ci si riferisce a caselle di controllo e pulsanti di opzione (radio button), vedrà attribuirsi un margine di 30 pixel.

### elem:not(selettore-di-esclusione)

Si applica a tutti gli elementi, a esclusione di quelli su cui agisce il selettore indicato tra parentesi.

#### **LISTATO 11.25** elem:not(selettore-di-esclusione). Un esempio

```
p:not(#prologo) {background: #efefef;}
```

Poiché la regola si applica a tutti i paragrafi, fatto salvo quello contrassegnato dall'attributo `id` con valore “`prologo`”, il primo paragrafo indicato nel Listato 11.26 non soddisfa tale regola e ne rimane escluso. Il secondo paragrafo, invece, ne sarà oggetto di applicazione.

#### **LISTATO 11.26** elem:not(selettore-di-esclusione). Caso di applicazione e di esclusione

```
<p id="prologo">Paragrafo escluso dalla regola poiché soddisfa il selettore di esclusione.</p>
<p>Paragrafo oggetto di applicazione della regola.</p>
```

### E ~ F

Questo selettore si applica all'elemento `F` se si trova dopo (ma non necessariamente subito dopo) l'elemento `E`, ed entrambi gli elementi condividono lo stesso elemento padre.

#### **LISTATO 11.27** E ~ F. Un esempio

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>E ~ F</title>
<style>
```

```

h2 ~ div.mumble {color: green; font-weight: bold;}

</style>

</head>
<body>
<h2>Titolo</h2>
<p>Un paragrafo</p>
<div class="mumble">L'elemento che soddisfa la regola.</div>
<h2>Test</h2>
<div>L'elemento non soddisfa la regola. Non dispone della classe mumble.</div>
</body>
</html>

```

Perché sia applicata questa regola devono essere soddisfatte una serie di condizioni:

- l'elemento `<h2>` e l'elemento `<div>` devono condividere lo stesso elemento contenitore;
- l'elemento `<div>` deve seguire, anche non immediatamente, l'intestazione di secondo livello `<h2>`;
- infine, l'elemento `<div>` deve vedersi attribuita una classe `mumble`.

Nel Listato 11.27 solo il primo elemento `<div>` risponde a tutti e tre questi requisiti.

## Riferimenti alle risorse citate e altri link utili

- Modulo CSS3 relativo ai selettori: <http://www.w3.org/TR/css3-selectors/>
- Una delle risorse più autorevoli sull'argomento, in lingua italiana, è Constile.org di Gianluca Troiani. Riguardo ai selettori CSS3 trovate un test cui sottoporre il vostro browser e il risultato di questa analisi all'indirizzo: <http://www.constile.org/cssselectorstest.html>
- L'importanza dei nuovi selettori è stata enfatizzata diverse volte da Eric Meyer, uno dei più importanti esperti sui fogli di stile, come per esempio in questa intervista rilasciata a Six Revision: <http://sixrevisions.com/interviews/six-questions-eric-meyer-on-css3/>
- Anche CSS3.info sottopone il vostro browser a un test sui selettori: <http://tools.css3.info/selectors-test/test.html>

## Conclusioni

I selettori introdotti con il modulo “Selectors Level 3” sono ampiamente supportati dai browser più diffusi; le lacune più rilevanti si concentrano nelle versioni più vecchie di

Internet Explorer. Il linguaggio esprime la sua forza proprio grazie a selettori così raffinati da poter identificare elementi arbitrariamente annidati anche in una pagina web con una struttura complessa. Inoltre, questi selettori contribuiscono in modo significativo a ridurre il grado di dipendenza esistente tra il documento che riceve gli stili e le regole di stile vere e proprie, rendendo più semplice riutilizzare entrambe le componenti.

# Web Fonts ed effetti tipografici

## Web Fonts

“A List Apart”, uno dei punti di riferimento per chiunque si occupi di tecnologie legate al Web, apre la sua sezione sui Web Fonts scrivendo “C’è vita dopo Georgia e Verdana”, sintetizzando in una frase così breve il motivo di tanta attenzione verso il modulo CSS relativo ai Web Fonts. Quest’ultimo promette di estendere in modo significativo le opzioni di scelta. Ancora oggi i font utilizzati sono prevalentemente Arial, Verdana, Helvetica, Georgia, Times, Courier. Grazie alla loro diffusione, tali font offrono la ragionevole certezza di trovarli disponibili nella stragrande maggioranza dei casi sui computer dei nostri visitatori.

Oggi per impostare il tipo di caratteri in cui sarà visualizzato il testo che pubblichiamo sul Web scriviamo una regola del tipo:

**LISTATO 12.1** La proprietà font-family

```
body {font-family: Arial, Helvetica, Sans-serif;}
```

In questo modo il browser cercherà di adottare il primo tipo di caratteri indicato nella lista (Arial, nel nostro caso). Se esso non è disponibile, si utilizzerà la seconda opzione (qui è il tipo di caratteri Helvetica); se anche questa non fosse disponibile, l’ultima opzione dovrebbe consentire il browser di selezionare un tipo di caratteri che, se non altro, appartenga alla stessa famiglia dei primi due.

La regola `@font-face` permette di includere un qualunque tipo di caratteri aggirando la limitazione legata al set di caratteri “a prova di Web”, ossia quelli che sono quasi certamente già installati sul computer di chi accede al nostro sito. Una rapida occhiata alla Tabella 12.1 mette in evidenza come il supporto sia sufficientemente diffuso.

**TABELLA 12.1** Supporto dei Web Fonts

ARGOMENTO	BROWSER
Web Fonts	Chrome 5+, Firefox 3.5+, IE4+, Opera 10.1+, Safari 3.1+

Prima di decidere di ricorrere ai Web Fonts possiamo testarne il supporto grazie a Modernizr. Se decidiamo di eseguire questo test al termine del caricamento della pagina, possiamo scrivere quanto appare nel Listato 12.2.

#### **LISTATO 12.2** Test del supporto dei Web Fonts per mezzo di Modernizr al termine del caricamento del documento

```
<script>
if (Modernizr.fontface) {
  // @font-face è supportato!
} else {
  // come strategia alternativa possiamo decidere
  // di caricare immagini in sostituzione al testo o
  // usare una libreria esterna come Cufon
}
</script>
```

Se invece vogliamo testare il supporto dei Web Fonts non al termine ma durante il caricamento della pagina, è preferibile utilizzare il codice proposto nel Listato 12.3.

#### **LISTATO 12.3** Test del supporto dei Web Fonts per mezzo di Modernizr durante il caricamento del documento

```
<script>
Modernizr._fontfaceready(function(bool) {
  if (bool) {
    // @font-face è supportato!
  } else {
    // si richiede strategia alternativa
  }
});
</script>
```

Vediamo qual è la sintassi di questa regola:

#### **LISTATO 12.4** Regola @font-face: un esempio

```
@font-face {
  font-family: MEgalopolisExtra;
  src: url(fonts/MEgalopolisExtra.otf);
}
```

La prima dichiarazione assegna un nome al set di caratteri. La seconda dichiarazione è

utile all'identificazione del file, con estensione `.otf`, a essi relativo. A questo punto il browser dispone di un tipo aggiuntivo di caratteri che possiamo invocare in una successiva dichiarazione standard del tipo di caratteri attraverso la proprietà `font-family`.

### **LISTATO 12.5** Applichiamo il set di caratteri che abbiamo scelto

```
body {font-family: MEgalopolisExtra, sans-serif;}
```

Nell'ipotesi in cui la regola `@font-face` non sia supportata sarà ignorata e, conseguentemente, il nome del tipo di caratteri che si è tentato di impostare mediante tale regola sarà scartato. La Tabella 12.1 sembra rassicurante nel documentare un supporto esteso alla regola `@font-face`, tuttavia l'esecuzione del Listato 12.6 riserva qualche sorpresa.

### **LISTATO 12.6** Regola `@font-face`

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>@font-face</title>
    <style>
      @font-face {
        font-family: MEgalopolisExtra;
        src: url(fonts/MEgalopolisExtra.otf);
      }
      body {font-family: MEgalopolisExtra, sans-serif; font-size: 26pt;}
    </style>
  </head>
  <body>
    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.</p>
  </body>
</html>
```

Scopriamo che Internet Explorer, in qualunque versione lo si consideri, ignora il font `MEgalopolisExtra`. Ciò è dovuto al fatto che diversi browser hanno privilegiato il supporto di formati diversi. La Tabella 12.2 aiuta a riepilogare la situazione.

### **TABELLA 12.2** Supporto dei formati di tipi caratteri

ESTENSIONE	FORMATO	SUPPORTO NEI PRINCIPALI BROWSER
.ttf	True Type	Chrome4+, Firefox3.5+, Opera 10+, Safari 3.1+,
.otf	Open Type	Chrome4+, Firefox3.5+, Opera 10+, Safari 3.1+
.eot	Embedded OpenType	IE4
.svg	SVG Font	Opera10+
.woff	Web Open Font Format	Chrome6+, Firefox3.6+, IE9

Questa tabella dimostra che, nell'ipotesi in cui il browser supporti la regola `@font-face`, diventa cruciale fornire il tipo di carattere nel formato corretto; inoltre, la Tabella 12.2 dimostra come non sia possibile servire un unico formato.

A questo punto però disponiamo solo del formato `.otf`, mentre per supportare adeguatamente anche chi utilizza Internet Explorer dovremmo dotarci di un file con estensione `.eot`. Font Squirrel (Figura 12.1) è un servizio online di conversione dei formati di carattere gratuito ed efficiente.



**FIGURA 12.1** Font Squirrel converte i formati dei caratteri e produce il codice da copiare e incollare nelle proprie pagine web per visualizzare il carattere prescelto.

Dovremo pensare di proporne almeno due se vogliamo coprire un'alta percentuale di utenti. Paul Irish, uno dei due creatori di Modernizr, ha elaborato la tecnica cross-browser che soddisfa questi vincoli e che è proposta nel Listato 12.7.

### **LISTATO 12.7** Tecnica di utilizzo cross-browser della regola @font-face

```
@font-face {  
    font-family: MEgalopolisExtra;  
    src: url(/fonts/my_web_licensed_font.eot);  
    src: local(MEgalopolisExtra), url(/fonts/my_web_licensed_font.ttf)  
    format("truetype");  
}  
  
.fontface {  
    font: 16px/24px MEgalopolisExtra, Helvetica, sans-serif;  
}  
  
.no-fontface { /*  
    strategia alternativa.  
*/ }
```

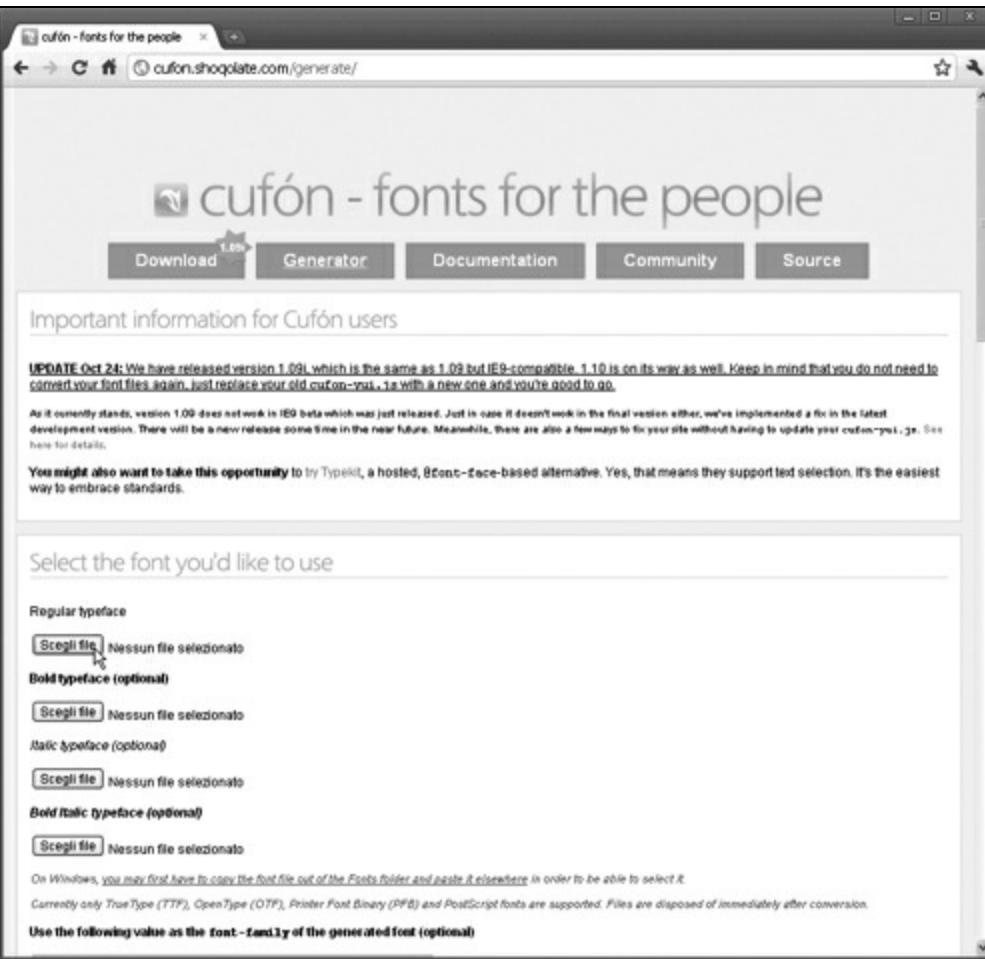
Per una breve spiegazione sul funzionamento di questa tecnica si rimanda al box seguente.

### Sintassi cross-browser della regola @font-face

Analizziamo dapprima le tre dichiarazioni di cui si compone la regola @font-face.

- `font-family: MEgalopolisExtra;` – Attribuiamo un nome a questo font in modo da poterlo assegnare successivamente alla proprietà `font-family` o `font`.
- `src: url (/fonts/my_web_licensed_font.eot);` – Questo è il percorso al file dei caratteri secondo il formato Embedded OpenType, come da Tabella 12.2. È l'unico formato che Internet Explorer (nelle versioni precedenti alla 9) sa interpretare, per cui provvederà a scaricare il file `my_web_licensed_font.eot`. Tutti gli altri browser ignoreranno questo formato.
- `src: local('My Web Font'), url(/fonts/my_web_licensed_font.ttf) format("truetype");` – Qui abbiamo due notazioni separate da virgola. Con `local(MEgalopolisExtra)` chiediamo al browser di cercare un tipo di carattere che presenti il nome `MEgalopolisExtra` tra quelli già installati su computer. Solo in caso negativo il browser procederà allo scaricamento del file in formato True Type. Internet Explorer, pur avendo già scaricato l'unico formato che sa gestire, tenterà di scaricare anche file in formato True Type. Tuttavia, sfruttando una lacuna di Explorer relativa all'incapacità di interpretare la direttiva sul formato, fallirà il suo tentativo di scaricare anche il file `.ttf`.

Disponendo di almeno un paio di formati di carattere diversi riusciamo a coprire un ampio spettro di browser. Tuttavia, nell'ipotesi più nefasta, quella cioè di una mancata implementazione della regola `@font-face`, abbiamo ancora un asso nella nostra manica che si chiama "Cufón". Cufón è un generatore di caratteri che si presenta rassicurante sin dallo slogan scelto: "fonts for the people".



**FIGURA 12.2** La home page di Cufón da cui partire per generare il file .js.

Attraverso una semplice procedura che possiamo eseguire online presso il suo sito (Figura 12.2), Cufón genera un file JavaScript partendo da un formato di caratteri `.ttf`, `.otf`, `.pfb` o PostScript che ci siamo premurati di fornire come punto di partenza di questo processo. Il file `.js` generato da Cufón sarà poi incluso nella pagina web e utilizzato per applicare il nostro font su qualunque browser.

#### **LISTATO 12.8** Visualizzazione di tipi di caratteri alternativi con Cufón

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Cufón in azione</title>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/
      jquery.min.js" type="text/javascript"></script>
    <script src="cufon-yui.js"></script>
    <script src="MEgalopolis.js"></script>
    <script>
      Cufon.replace('h1');
      Cufon.replace('#sub1');
    </script>
```

```

</head>
<body>
  <h1>Yes you can</h1>
  <h2 id="sub1">Utilizza tipi di caratteri decisamente
  originali!</h2>
  <script> Cufon.now(); </script>
</body>
</html>

```

Cufón necessita di una serie di file JavaScript per funzionare correttamente:

- `cufon-yui.js`, il file che rappresenta il cuore del generatore di caratteri;
- `MEgalopolis.js`, il file che Cufón ha prodotto sulla base del file rappresentante un formato di caratteri tra quelli accettati;
- jQuery, Sizzle, MooTools, Dojo, Prototype o qualunque altra libreria su cui Cufón possa fare affidamento per applicare stili riconoscibili anche attraverso selettori ID (come nell'esempio `#sub1`) o selettori di classe su Internet Explorer 6 e 7.

Il Listato 12.8 riproduce il tipo di carattere MEgalopolis anche su Konqueror 4.1.14 (Figura 12.3).



**FIGURA 12.3** Konqueror 4.1.14 è uno dei browser che beneficiano di Cufón.

Mettiamo insieme tutti i pezzi di questo mosaico nel Listato 12.9 e otteniamo un codice con il quale siamo in grado di offrire a un ampio numero di browser un tipo di caratteri che va oltre quelli definiti "a prova di Web".

**LISTATO 12.9** Web Fonts con regola @font-face e strategia alternativa basata su Cufón

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>@font-face</title>
    <script src="modernizr-1.6.min.js"></script>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2
    /jquery.min.js" type="text/javascript"></script>

```

```

<script src="cufon-yui.js" type="text/javascript"></script>

<script>
Modernizr._fontfaceready(function(bool) {
    // se @font-face non è supportato può essere utile usare Cufón
    if (!bool) getScript('cufon.withfont.min.js',function(){
        Cufon.now();
    });
});
</script>
<style>
@font-face {
    font-family: MEgalopolisExtra;
    src: url(fonts/megalopolisextra-webfont.eot);
    src: local('MEgalopolisExtra'),
        url(fonts/megalopolisextra-webfont.ttf) format("truetype");
}
.fontface {
    font: 16px/24px MEgalopolisExtra, serif;
}
.no-fontface {
    font: 16px/24px serif;
}
</style>
</head>
<body>
    <h1>Yes You Can</h1>
    <h2 id="sub1">Utilizza tipi di caratteri decisamente originali!</h2>
    <script>
Modernizr._fontfaceready(function(bool) {
    // se @font-face non è supportato ci si affida a Cufón
    if (!bool) getScript('MEgalopolis.js',function(){
        Cufon.now();
    });
});
</script>
</body>
</html>

```

## NOTA

Questa soluzione finale, sebbene abbia il vantaggio di coprire un insieme sufficientemente esteso di browser, finisce con il generare una forte dipendenza da JavaScript che, se introdotto solo ed esclusivamente a questo fine, sembra troppo penalizzante in termini di performance rispetto all'alternativa delle immagini a sostituzione del testo.

## Effetti tipografici

## Testo ombreggiato

Le ombreggiature applicate al testo (Figura 12.4) sono, non da oggi, uno degli effetti tipografici più diffusi.

A box containing the text "Lorem ipsum dolor sit amet" in a bold, sans-serif font. The text is partially obscured by a dark gray drop shadow effect, giving it a 3D appearance as if it were floating or casting a shadow.

**FIGURA 12.4** L'ombreggiatura conferisce profondità.

Si tratta di visualizzare una copia del testo che, rispetto all'originale, presenta un certo grado di sfocatura e che appaia parzialmente scostato. In questo modo si ottiene un effetto di profondità del testo, che sembra proiettare un'ombra. Di norma questo risultato si ottiene per mezzo di un software di elaborazione delle immagini. La promessa della proprietà `text-shadow` è quella di liberarci dall'onere della creazione dell'immagine: lo stesso effetto risulterà applicabile direttamente sul testo, con il duplice vantaggio di realizzare minori connessioni HTTP (ogni elemento incluso nel documento ne richiede una) e ottenere al contempo una pagina web più leggera.

**TABELLA 12.3** Supporto del testo ombreggiato

PROPRIETÀ	BROWSER
<code>text-shadow</code>	Chrome 2+, Firefox 3.5+, Opera 9.5+, Safari 3+

Il Listato 12.10 illustra la sintassi di una regola CSS che applica l'ombreggiatura e produce il risultato apprezzabile in Figura 12.5.

**LISTATO 12.10** Stile di ombreggiatura

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Ombreggiatura</title>
<style>
body { margin: 3em; }
h1 {text-shadow: 3px 4px 2px #888;}
</style>
</head>
<body>
```

```
<h1>Un classico effetto ombreggiatura</h1>
```

```
</body>
```

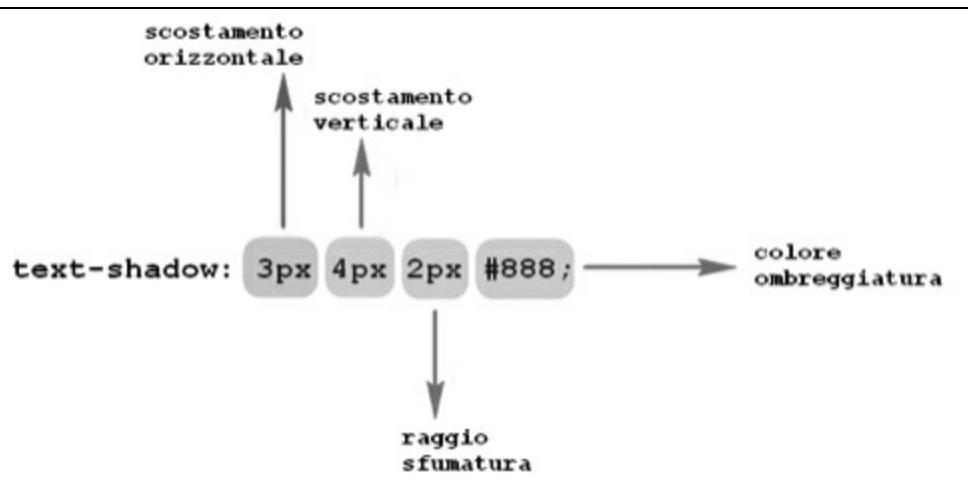
```
</html>
```

## Un classico effetto ombreggiatura

**FIGURA 12.5** Testo con ombreggiatura.

A questa proprietà assegniamo quattro valori, separati da spazio:

- il primo rappresenta lo scostamento orizzontale del testo ombreggiato rispetto a quello originale (nell'esempio è impostato a 3 pixel);
- il secondo imposta lo scostamento verticale del testo ombreggiato rispetto a quello originale (nell'esempio questo valore è uguale a 4 pixel);
- il terzo esprime quanto deve essere sfumato il testo per mezzo della lunghezza di un raggio di sfumatura (nell'esempio questo valore è impostato a 2 pixel);
- l'ultimo parametro rappresenta il colore del testo ombreggiato, qui impostato a una tonalità di grigio scuro (Figura 12.6).



**FIGURA 12.6** Analisi dei quattro argomenti che definiscono l'ombreggiatura del testo.

In realtà, giocando con questi valori scopriamo che questa tecnica è alla base di altri effetti tipografici: bagliore (Listato 12.11 e Figura 12.7), sfocatura (Listato 12.12 e Figura 12.8).

### **LISTATO 12.11** Stile di ombreggiatura

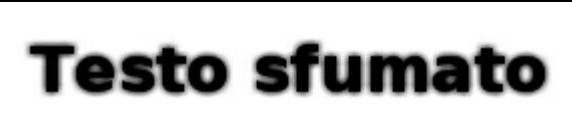
```
body {background-color: #000;}  
h1 {text-shadow: 1px 1px 5px #fff;}
```

# Effetto bagliore

**FIGURA 12.7** Ogni lettera sembra retro-illuminata da una soffusa luce bianca.

## LISTATO 12.12 Sfocatura (altresì detto effetto blur)

```
h1 {text-shadow: 0px 0px 5px #000;}
```



Testo sfumato

**FIGURA 12.8** L'ombra si trova così a ridosso del testo cui si riferisce da renderlo sfumato.

## NOTA

Se il valore degli scostamenti orizzontale e verticale è pari a zero, ne consegue che l'ombra è posta alle spalle del testo, generando in tal modo un effetto sfumatura.

## Ombreggiature multiple

Sin qui abbiamo appreso come conferire l'ombreggiatura al testo, tuttavia più regole possono essere applicate contemporaneamente allo stesso testo avendo cura di separare ogni set di quattro argomenti da una virgola, come nel Listato 12.13.

## LISTATO 12.13 Ombreggiature multiple

```
text-shadow: 0 0 4px #000, 0 -5px 4px #ff3, 2px -10px 6px #fd3, -2px -15px 11px #f80, 2px -25px 18px #f20;
```

È proprio sfruttando questa opportunità che possiamo riprodurre l'effetto (improbabile) di una serie di lettere che stanno andando a fuoco.



Le lettere vanno a fuoco

**FIGURA 12.9** Un uso scenografico dell'effetto di ombreggiatura.

## NOTA

È necessario tenere a mente che valori negativi dello scostamento orizzontale posizionano l'ombra a sinistra del testo. Valori negativi dello scostamento verticale posizionano l'ombra al di sopra del testo.

## Word Wrap (come spezzare una parola)

La proprietà `word-wrap` nasce con lo scopo di evitare che una parola di più sillabe possa estendersi al di fuori dell'area in cui altrimenti sarebbe stata confinata. La Tabella 12.4 sintetizza il supporto della proprietà presso i principali browser.

**TABELLA 12.4** Supporto della proprietà `word-wrap`

PROPRIETÀ	BROWSER
<code>word-wrap</code>	Chrome, Firefox 3.5+, IE5.5+, Opera 10.5+, Safari

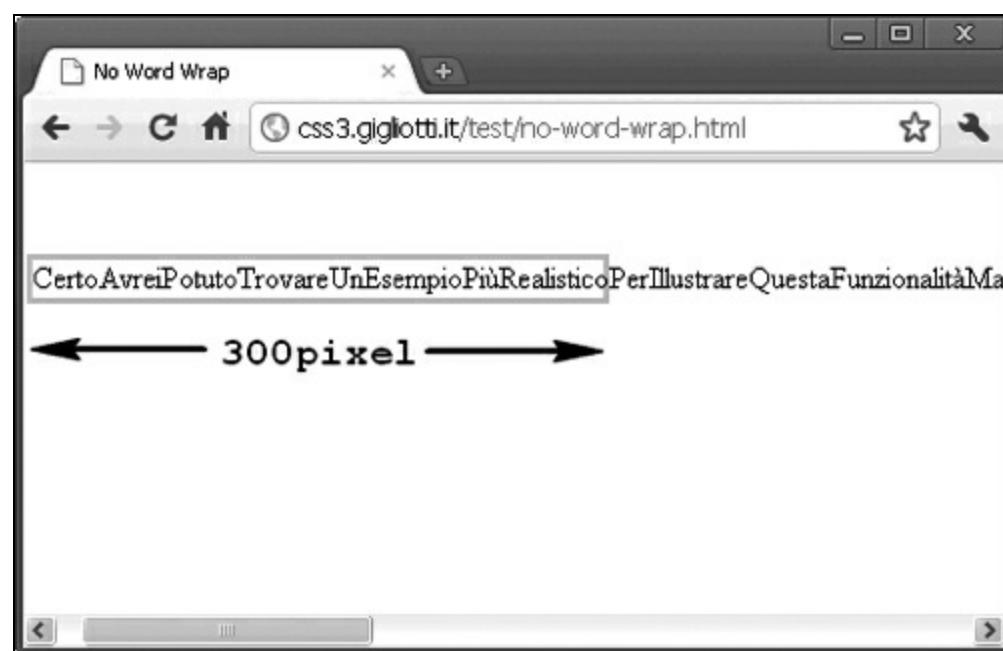
L'esempio, seppure surreale, è utile a enfatizzare il funzionamento di questa proprietà.

**LISTATO 12.14** Parole troppo lunghe. Un esempio

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Word Wrap</title>
    <style>
      body { margin: 3em; }
      div {
        border: 3px solid rgb(200, 200, 180);
        width: 300px;
      }
    </style>
  </head>
  <body>
    <div
      id="ritornoacapo">CertoAvreiPotutoTrovareUnEsempioPiùRealisticoPerIllustrareQuestaFunzionalitàMaNonTrovon
      aEssereUtileApplicareUnRitornoACapoSeNecessarioAncheInUnPuntoArbitrarioDellaParola</div>
  </body>
</html>
```

Come risulta dalla Figura 12.10, una parola troppo lunga si colloca parzialmente al di fuori dall'area assegnata all'elemento che dovrebbe contenerla: la barra di scorrimento orizzontale è conseguenza di un testo privo di interruzioni che si protrae ben oltre i 300 pixel di larghezza del contenitore `<div>`. Per porre rimedio a questo scenario si fa ricorso

alla proprietà `word-wrap` in modo da avere una disposizione del testo nell'elemento `<div>` che sia composto come in Figura 12.11.



**FIGURA 12.10** Il testo occupa più spazio di quello assegnato all'elemento contenitore. La barra di scorrimento orizzontale enfatizza lo scenario.



**FIGURA 12.11** Per quanto lunga, la parola viene "costretta" nel suo contenitore.

La proprietà `word-wrap` ammette due possibili valori: `normal`, che ne rappresenta il valore predefinito ed è alla base del comportamento ordinario (le parole non sono mai interrotte per cambiare righe), e `break-word`, che all'occorrenza va a capo introducendo un'interruzione arbitraria nel corso della parola stessa.

text-overflow

**TABELLA 12.5** Supporto della proprietà text-overflow

PROPRIETÀ	BROWSER
text-overflow	Chrome2+, IE6+, Opera10.7, Safari1.3+

Nel caso in cui un testo sia talmente lungo da proiettarsi al di fuori del suo elemento contenitore è possibile applicare i tre puntini di sospensione nel punto esatto in cui il testo esaurisce tutto lo spazio per esso allocato.

**LISTATO 12.15** Proprietà text-overflow. Un esempio

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Proprietà: text-overflow</title>
    <style>
      body { margin: 3em; }
      div {
        border: 3px solid rgb(200, 200, 180);
        width: 300px;
        overflow: hidden;
        white-space: nowrap;
        -ms-text-overflow: ellipsis;
        -o-text-overflow: ellipsis;
        -webkit-text-overflow: ellipsis;
        text-overflow: ellipsis;
      }
    </style>
  </head>
  <body>
    <div id="ritornoacapo">Certo avrei potuto trovare un esempio più realistico per illustrare questa funzionalità</div>
  </body>
</html>
```

Certo avrei potuto trovare un esempio più reali...

**FIGURA 12.12** Il testo eccede l'area su cui si estende l'elemento <div> e questo costringe ad applicare i tre puntini di sospensione.

Per comprendere come sia stato possibile riprodurre questo risultato può essere utile

analizzare anche altre proprietà cui è stato necessario fare ricorso. Innanzitutto, per ottenere questo effetto abbiamo avuto cura di assegnare il valore `hidden` alla proprietà `overflow`. Questa proprietà agisce sulla modalità di visualizzazione del contenuto qualora questo travalichi i confini dell'elemento contenitore. Il suo valore predefinito è visibile, il che implica che testo e altri elementi siano visualizzati anche se si estendono al di fuori dell'area demarcata dall'elemento contenitore. Il valore `hidden`, di fatto, rovescia questo comportamento nascondendo quella parte del contenuto che fuoriesce dal suo elemento padre.

Un'altra proprietà su cui è necessario intervenire per ottenere questo risultato è `white-space` al valore `nowrap`, che obbliga il browser a non andare a capo in presenza di uno spazio; in altri termini, questo valore lo porta a derogare dal suo comportamento abituale che si ottiene quando `white-space` è impostato a `normal`.

## Impaginazione del testo in colonne

Ottenere una fluida impaginazione del testo in colonne, a somiglianza di quanto accade nei quotidiani: è, in sintesi, l'obiettivo di un intero modulo CSS3 chiamato "Multi-column layout". Si tratta di un altro strumento a nostra disposizione per permetterci di discostarci definitivamente dalle tabelle HTML per meri fini di layout della pagina web. Abbandonare una tecnica che abbiamo appreso, magari con fatica, nel corso degli anni, può sembrare poco allettante, ma vediamo subito con quali benefici sarà ripagato il nostro sforzo di apprendimento dell'impaginazione del testo in colonne secondo le proprietà elencate in questo modulo.

- Si ottiene un codice HTML più pulito e leggero. Questo è già un ottimo risultato, perché il documento pesa di meno: lo stesso risultato si raggiunge con un minor numero di kilobyte. Un documento HTML più semplice equivale a una sua rappresentazione a oggetti (DOM) più agile e semplifica il lavoro del codice JavaScript e dei fogli di stile.
- Il testo scorre in modo fluido da una colonna all'altra adattandosi a ogni ridimensionamento dell'elemento contenitore. Questo semplicemente non è possibile utilizzando le tabelle per definire la struttura di un documento: il testo in questo caso rimarrebbe ingabbiato nella cella in cui è stato inserito.
- Gli screen-reader (come Jaws, per esempio) operano con difficoltà la lettura del testo che si interrompe in punti arbitrari per poi riprendere in una cella diversa di una tabella. La suddivisione del testo in colonne è solo una rappresentazione visuale di informazioni contrassegnate con il ricorso a marcatori HTML che sono lasciati in grado di assicurare una migliore rappresentazione semantica e conseguentemente una più facile lettura anche da parte di screen-reader.

Il supporto di questo modulo è espresso in Tabella 12.6.

**TABELLA 12.6** Supporto del testo in colonne

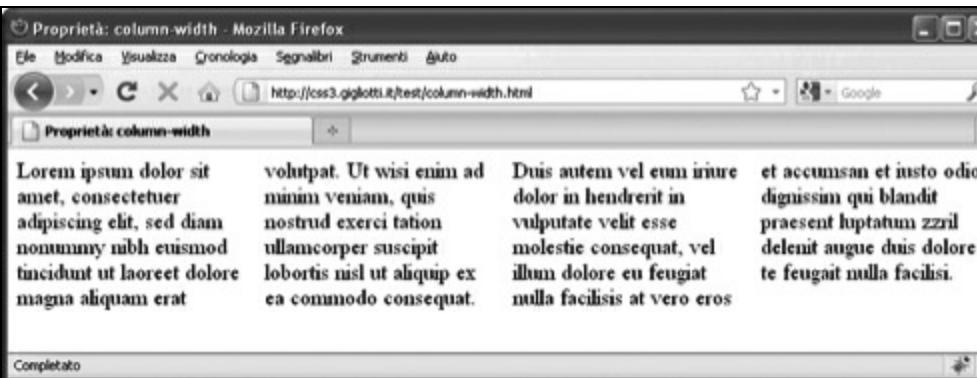
ARGOMENTO	BROWSER
Colonne multiple*	Chrome4+, Firefox3+, Safari4+

\* Il supporto è ancora conseguito con impiego dei prefissi proprietari `-webkit-` (per Chrome e Safari) e `-moz-` (per Firefox).

Suddividere in più colonne il testo presente all'interno di uno stesso elemento contenitore è un'operazione semplice. Il Listato 12.16 e la successiva Figura 12.13 illustrano un'applicazione pratica.

**LISTATO 12.16** Testo in colonne. Proprietà `column-width`

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Proprietà: column-width</title>
    <style>
      div {
        -moz-column-width: 10em;
        -webkit-column-width: 10em;
        column-width: 10em;
      }
    </style>
  </head>
  <body>
    <div>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut
    laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper
    suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in
    vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et
    iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.
    </div>
  </body>
</html>
```



**FIGURA 12.13** Testo in colonne di 10em di larghezza.

## NOTA

Il presente stato di implementazione presso i principali browser prevede il ricorso alle proprietà con suffissi proprietari. I suffissi `-moz` e `-webkit` qui sono necessari per abilitare la funzionalità su Mozilla, Chrome e Safari. Internet Explorer non riconosce questa proprietà.

La proprietà `column-width` accetta una misura, per esempio in pixel o em, attraverso la quale è possibile impostare la larghezza minima della colonna: nel listato si definisce un'ampiezza minima pari a `10em` che, per un tipo di caratteri con grazie come il Times New Roman e per 12 punti di dimensione del carattere, corrisponde circa a 5 parole per riga. Poiché nel Listato 12.16 il marcitore `<div>` che rappresenta l'elemento contenitore del testo assumerà dimensioni diverse al ridimensionamento della finestra, ne consegue che riducendo la finestra del browser il numero di colonne si adatterà alle nuove dimensioni dell'elemento contenitore (Figura 12.14).

Una modalità alternativa di impostare questo vincolo consiste nel ricorrere alla proprietà `column-count`. Nell'esempio precedente avremmo potuto scrivere:

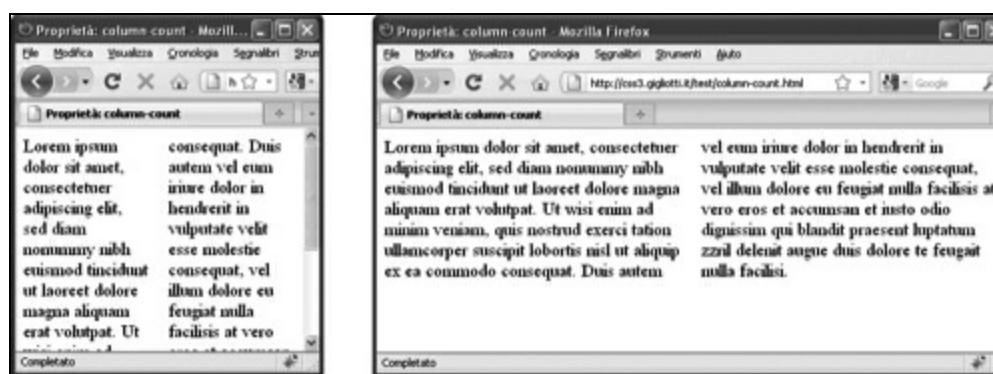
## LISTATO 12.17 Testo in colonne: la proprietà column-count

```
<style>
div {
  -moz-column-count: 2;
  -webkit-column-count: 2;
  column-count: 2;
}
</style>
```



**FIGURA 12.14** Il numero delle colonne si riduce, nel rispetto del vincolo della larghezza minima dei 10em.

In questo caso avremo una visualizzazione del testo rigorosamente ancorata alle due colonne, indipendentemente dalle dimensioni dell'elemento contenitore (Figura 12.15).



**FIGURA 12.15** Il numero di colonne rimane inalterato pur modificando le dimensioni dell'elemento contenitore.

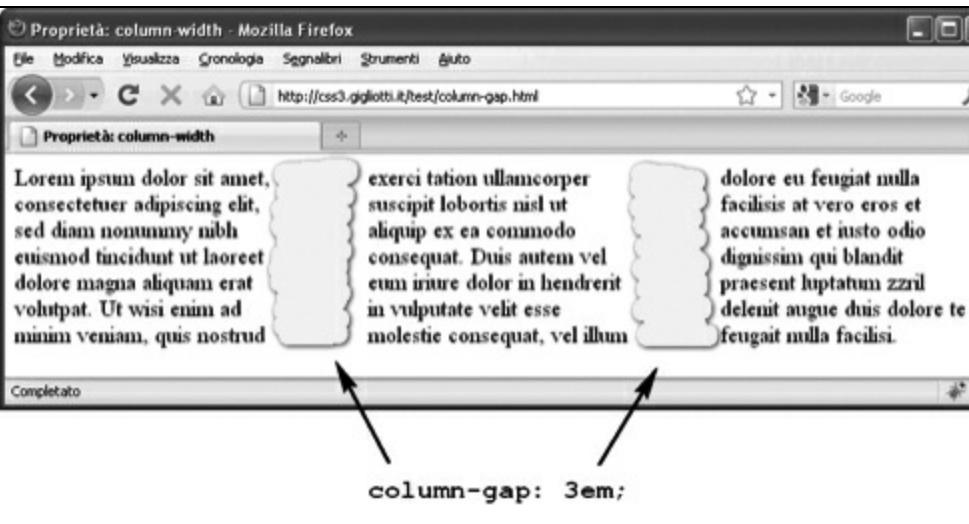
La proprietà `columns` rappresenta una forma di notazione contratta attraverso la quale possiamo impostare tanto il numero di colonne che la loro dimensione.

#### **LISTATO 12.18** Colonne multiple: notazione contratta

```
div {columns: 2;}  
div {columns: 10em;}  
div {columns: 2 10em;}
```

Nel primo caso si imposta il numero di colonne, nel secondo la dimensione minima di larghezza e nel terzo entrambe.

Nelle ultime immagini avrete notato come lo spazio esistente tra le colonne adiacenti sia rimasto in ogni caso inalterato. Attraverso la proprietà `column-gap` possiamo intervenire sulla dimensione di quest'area, che la specifica suggerisce di mantenere al valore predefinito (pari a `1em` qualora non sia esplicitamente specificato un valore diverso).



**FIGURA 12.16** Valori più alti di column-gap aumentano lo spazio tra colonne adiacenti comprimendo lo spazio per il testo al loro interno.

L'effetto ottenuto in Figura 12.16 è il risultato della regola seguente:

**LISTATO 12.19** column-gap interviene sullo spazio esistente tra colonne adiacenti

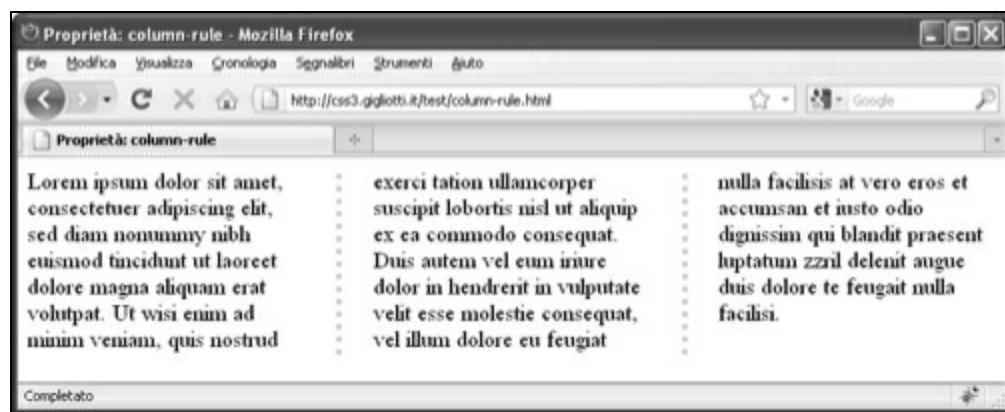
```
div {
  -moz-column-width: 10em;
  -webkit-column-width: 10em;
  column-width: 10em;
  -moz-column-gap: 3em;
  -webkit-column-gap: 3em;
  column-gap: 3em;
}
```

Nel mezzo di quell'area su cui interviene la proprietà `column-gap` è posta una riga su cui agisce `column-rule`. Se al Listato 12.20 aggiungiamo quanto segue:

**LISTATO 12.20** La proprietà column-rule. Un esempio

```
div {
  -moz-column-width: 10em;
  -webkit-column-width: 10em;
  column-width: 10em;
  -moz-column-gap: 3em;
  -webkit-column-gap: 3em;
  column-gap: 3em;
  -moz-column-rule: 0.3em dotted #ccc;
  -webkit-column-rule: 0.3em dotted #ccc;
  column-rule: 0.3em dotted #ccc;
}
```

Otteniamo il risultato illustrato in Figura 12.17.



**FIGURA 12.17** La riga posta tra le colonne è sotto il controllo della proprietà `column-rule`.

È da notare che questa riga non incide in alcun modo sullo spazio delle colonne e dello spazio che le divide. In seconda battuta si osserva che la proprietà `column-rule` rappresenta una forma contratta di tre diverse proprietà.

- `column-rule-width`: accetta come valore una misura della larghezza della riga.
- `column-rule-style`: interviene sulla rappresentazione grafica della riga. Nel Listato 12.20 si è utilizzato il termine `dotted` per ottenere il risultato di Figura 12.17, ma avrei potuto impiegare uno qualunque degli altri 9 valori impiegati per la proprietà `border-style: none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset`.
- `column-rule-color`: definisce il colore in cui sarà visualizzata la riga di separazione delle colonne.

## Riferimenti alle risorse citate e altri link utili

- La specifica del modulo CSS Fonts Level 3: <http://www.w3.org/TR/css3-fonts>
- La specifica del modulo CSS Text Level 3: <http://www.w3.org/TR/css3-text/>
- La specifica del modulo CSS Web Fonts: <http://www.w3.org/TR/css3-webfonts>
- “A List Apart” dedica all’argomento dei Web Fonts una serie di articoli, tutti consultabili online all’indirizzo: <http://www.alistapart.com/topics/design/web-fonts/>
- Il tipo di caratteri utilizzato nell’esempio relativo ai Web Fonts è stato scaricato da questo indirizzo: <http://www.smeltery.net/fonts/megalopolis-extra>
- Il sito <http://webfonts.info> si propone come valido punto di riferimento per l’argomento del Web Fonts.
- Un generatore di font in formati diversi disponibile on-line all’indirizzo: <http://www.fontsquirrel.com/fontface/generator>

- La strategia cross-browser di applicazione della regola `@font-face` che a oggi risulta migliore sotto il profilo delle prestazioni è stata ideata da Paul Irish ed è stata descritta a questo indirizzo: <http://paulirish.com/2009/bulletproof-font-face-implementation-syntax/>
- Scrivete “Cufón” e leggete “la vostra strategia alternativa per la visualizzazione di font decisamente originali”. La documentazione è disponibile presso: <https://github.com/sorccu/cufon/wiki/>
- Articolo introduttivo pubblicato su “A List Apart” sull’argomento del testo in colonne: <http://www.alistapart.com/articles/css3multicolumn>

## Conclusioni

In questo capitolo abbiamo evidenziato alcune delle novità di facile impiego relative ai moduli CSS di terzo livello di testo e colonne multiple. Nel prossimo capitolo ci si concentrerà sul modulo relativo a bordi e sfondi; scopriremo che con poche semplici regole è possibile ottenere gli stessi risultati che ancora oggi costano fatica e un codice poco manutenibile.

1

## Bordi e sfondi

Bordi

Un unico modulo di CSS3 si occupa della definizione di bordi e sfondi. In questo capitolo analizzeremo diverse proprietà definite in questo modulo a partire da `border-radius`; la Tabella 13.1 ne evidenzia il supporto e la successiva Figura 13.1 ne propone il risultato.

**TABELLA 13.1** Supporto delle proprietà border-\* -radius

<b>PROPRIETÀ</b>	<b>BROWSER</b>
border-radius	Chrome5+, Firefox 3.5+*, IE9, Opera10.5+ Safari 5+

\* Firefox richiede l'estensione proprietaria `-moz-` mentre le versioni di Safari precedenti alla 5 supportano la funzionalità solo con l'adozione del prefisso `-webkit-`.

Lorem ipsum dolor sit amet,  
  consectetuer adipiscing elit, sed diam  
  nonummy nibh euismod tincidunt ut  
  laoreet dolore magna aliquam erat  
  volutpat. Ut wisi enim ad minim  
  veniam, quis nostrud exerci tation  
  ullamcorper suscipit lobortis nisl ut  
  aliquip ex ea commodo consequat.

**FIGURA 13.1** Angoli arrotondati con la proprietà border-radius

Per ottenere questo risultato è sufficiente impostare la proprietà `border-radius`.

**LISTATO 13.1** Proprietà border-radius: un esempio

```
<!DOCTYPE html>  
<html>  
  <head>
```

```

<meta charset="utf-8">
<title>Ombreggiatura</title>
<style>
body { margin: 3em; background-color: #ccc; }
div {border-radius: 20px; background-color: #fff; width: 300px;
padding: 1em; }
</style>
</head>
<body>
<div>Lorem ipsum dolor sit amet [omissis]</div>
</body>
</html>

```

Il valore assegnato a questa proprietà esprime la lunghezza del raggio della sezione di cerchio necessaria a riprodurre l'angolo arrotondato. Questo è un effetto che ha sempre richiesto, in passato, un utilizzo distorto del linguaggio di marcatura e la creazione di tabelle per produrre risultati simili a quelli di Figura 13.1. Per avere un'idea di quanto questa proprietà possa aiutare in termini di pulizia del linguaggio di marcatura, si presenta nel Listato 13.2, a titolo di paragone, la soluzione adottata finora per rendere possibile questo effetto.

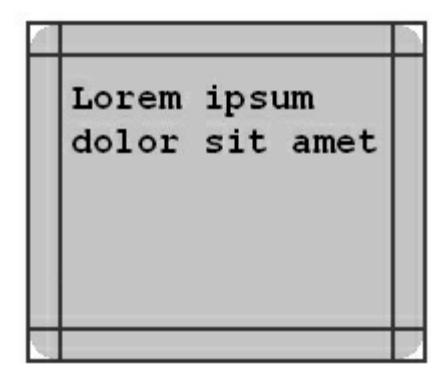
### **LISTATO 13.2** La "vecchia" tecnica impiegata per creare angoli arrotondati

```

<table width="200" cellpadding="0" cellspacing="0" border="0">
<tr>
<td width="14"></td>
<td width="172"></td>
<td width="14"></td>
</tr>
<tr>
<td></td>
<td><p> Lorem ipsum dolor sit amet </p></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</table>

```

Per rendere evidente il perché di questa struttura, in Figura 13.2 ho evidenziato i bordi della tabella così riprodotta.

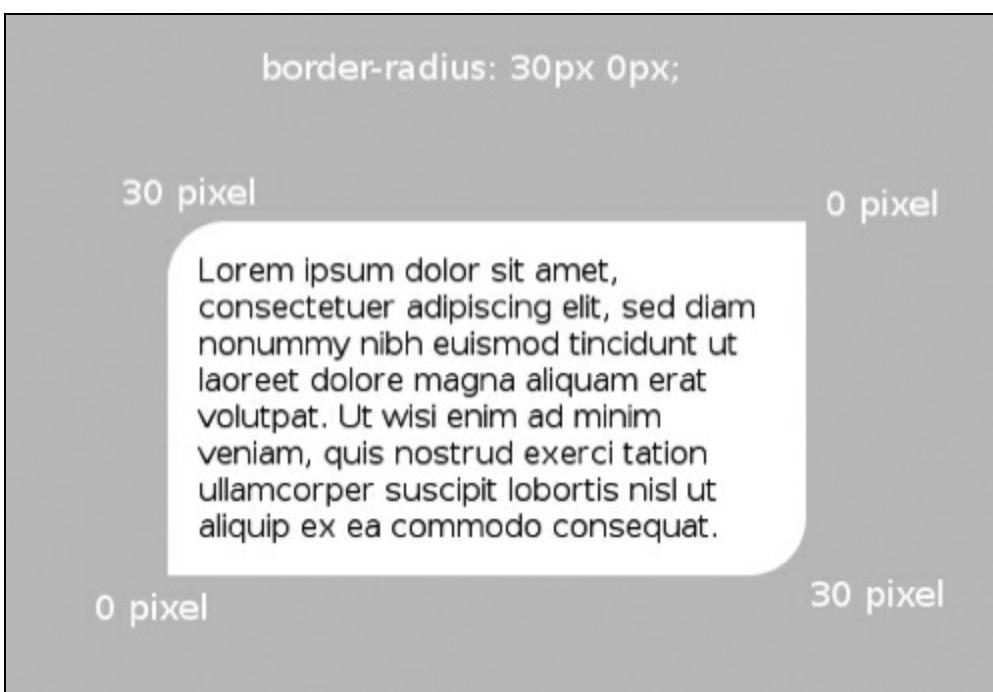


**FIGURA 13.2** Angoli arrotondati prima dell'avvento della proprietà border-radius.

Oltre alla quantità addizionale di elementi di marcatura, sulla quale è inutile soffermarsi, bisogna sottolineare che questa tecnica prevede, tra l'altro, la necessità di scaricare quattro immagini, una per ciascun angolo, dunque quattro richieste HTTP addizionali.

La proprietà `border-radius` è una proprietà sintetica che definisce, con il ricorso a un unico valore, i bordi arrotondati di tutti e quattro gli angoli. Tuttavia la stessa proprietà accetta uno, due, tre o quattro valori.

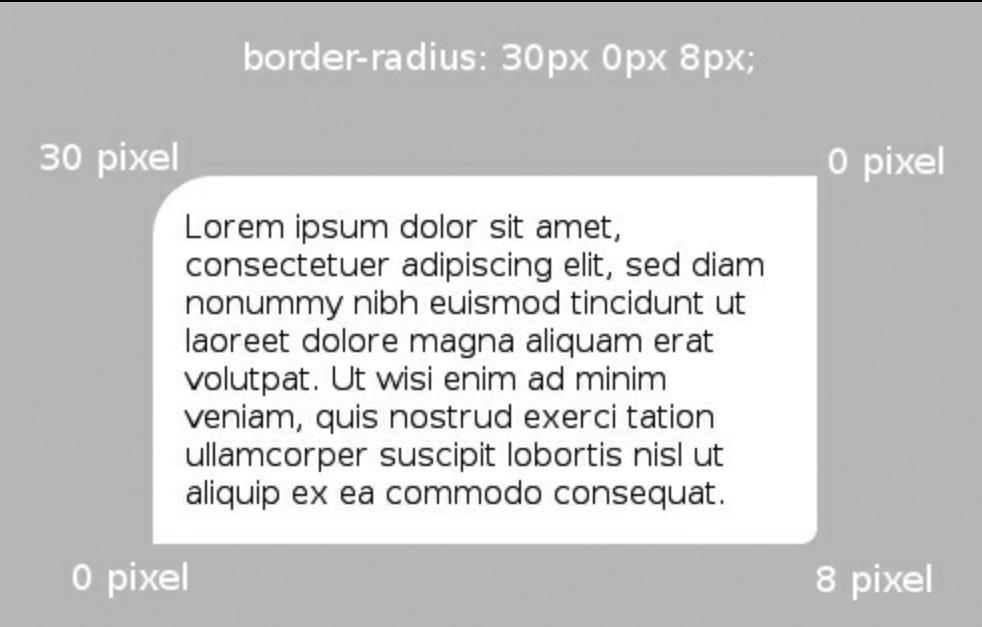
- Un valore: la stessa lunghezza del raggio è applicata a tutti e quattro gli angoli (Figura 13.1).
- Due valori: il primo valore impone il raggio del bordo in alto a sinistra e in basso a destra. Il secondo impone il raggio del bordo in alto a destra e in basso a sinistra (Figura 13.3). Un valore pari a zero equivale a definire uno spigolo.



**FIGURA 13.3** Effetto prodotto con "border-radius: 30px 0px;".

- Tre valori: il primo valore impone il raggio del bordo in alto a sinistra, il secondo impone il raggio dell'angolo in alto a destra e in basso a sinistra. Il terzo valore è

relativo all'angolo in basso a destra (Figura 13.4).



**FIGURA 13.4** Effetto prodotto con "border-radius: 30px 0px 8px;".

- Quattro valori. Ogni valore identifica un angolo diverso partendo da quello in alto a sinistra e procedendo in senso orario (Figura 13.5).



**FIGURA 13.5** Effetto prodotto con “border-radius: 30px 70px 10px 0px;”.

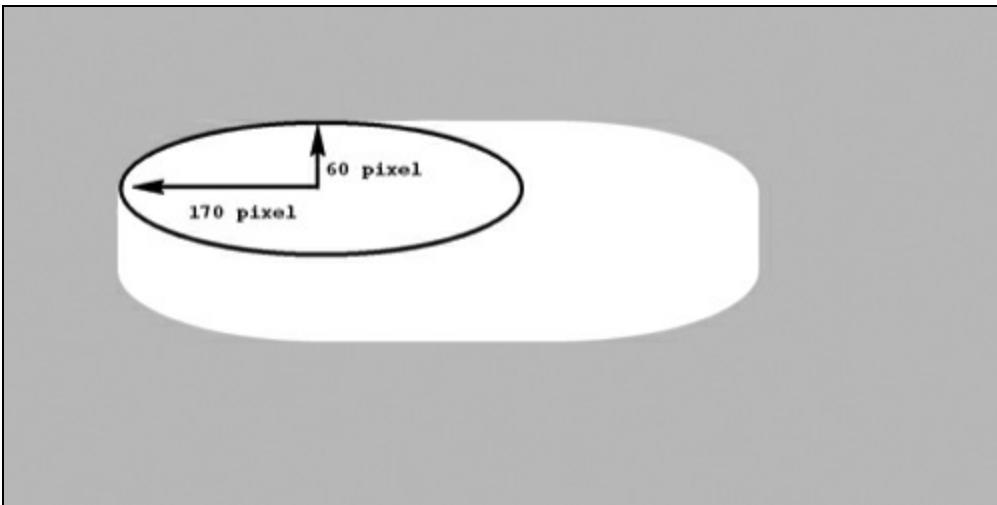
Finora ci siamo giovati di un’ulteriore semplificazione. Abbiamo ipotizzato che si potesse impostare il valore di un unico raggio per la definizione dell’arrotondamento. In realtà è possibile definire tanto la lunghezza di un raggio orizzontale quanto quella di un raggio verticale, avendo cura di separare i due valori con uno slash “/”.

### **LISTATO 13.3** Come definire diversi valori per il raggio orizzontale e quello verticale

border-radius: 170px/60px;

Questo produce il risultato apprezzabile in Figura 13.6, in cui una ellisse con bordo nero è stata parzialmente sovrapposta all'angolo posto più in alto a sinistra per illustrare l'impatto di questa sintassi dal punto di vista grafico.

Possiamo ora disfarcici della semplificazione introdotta in principio quando si è scritto del raggio di un cerchio. In realtà si tratta di un'ellisse: i raggi sono due, solo nel caso in cui coincidano l'angolo viene rappresentato come sezione di un cerchio, in tutti gli altri casi l'angolo arrotondato è la rappresentazione di una sezione di ellisse come chiaramente dimostra la Figura 13.6.



**FIGURA 13.6** Impostazione della lunghezza dei due raggi dell'ellisse con sintassi border-radius: lunghezza-raggio-orizzontale/lunghezza-raggio-verticale.

Finora abbiamo sempre definito la dimensione del raggio, sia esso verticale o orizzontale, solo in termini assoluti – ciò per mezzo della misura `px` – ma è possibile impostare lo stesso valore anche mediante valori relativi espressi in percentuali.

### **LISTATO 13.4** Formula espressa in valori percentuali

Border-radius: 5%/40%;

Un'altra piccola semplificazione di cui ci siamo giovati è che la proprietà `border-radius` permette di definire, secondo una sintassi breve, le stesse informazioni che altrimenti andrebbero applicate per mezzo di quattro distinte proprietà: `border-bottom-left-radius`, `border-bottom-right-radius`, `border-top-left-radius`, `border-top-right-radius`. Per quanto sin qui scritto si desume che le due regole elencate nel Listato 13.5 sono due modi diversi di esprimere le stesse regole.

### **LISTATO 13.5** `border-radius`. Due diverse sintassi che esprimono lo stesso insieme di regole

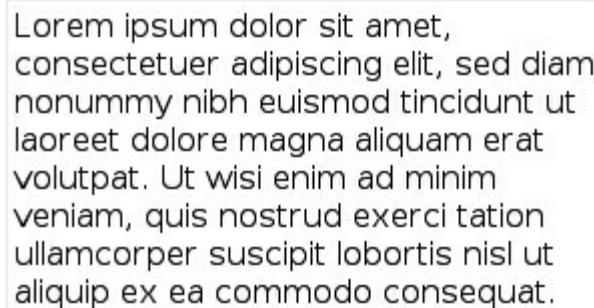
```
div {border-radius: 30px 10px 20px / 8px 12px;}  
div {  
    border-top-left-radius: 30px 8px;  
    border-top-right-radius: 10px 12px;  
    border-bottom-right-radius: 20px 8px;  
    border-bottom-left-radius: 10px 12px;  
}
```

## NOTA

Lo slash, opzionale nella proprietà `border-radius`, non è previsto per le singole proprietà `border-n-radius`.

## Ombreggiatura applicata ai blocchi: box-shadow

La proprietà `box-shadow` consente di applicare un'ombreggiatura a un elemento HTML come rappresentato in Figura 13.7.



Placeholder text inside the div:

Lorem ipsum dolor sit amet,  
consectetuer adipiscing elit, sed diam  
nonummy nibh euismod tincidunt ut  
laoreet dolore magna aliquam erat  
volutpat. Ut wisi enim ad minim  
veniam, quis nostrud exerci tation  
ullamcorper suscipit lobortis nisl ut  
aliquip ex ea commodo consequat.

**FIGURA 13.7** Ombreggiatura applicata a un elemento <div>.

**TABELLA 13.2** Supporto della proprietà `box-shadow`

PROPRIETÀ	BROWSER
<code>box-shadow</code>	Chrome 4+*, Firefox 4, IE9, Opera 10.5+, Safari 3+*

\* Chrome e Safari necessitano del suffisso `-webkit-` mentre per le versioni di Firefox precedenti alla 4 è necessario utilizzare il prefisso `-moz-`.

La proprietà `box-shadow`, di cui la Tabella 13.2 ricorda il supporto, accetta quattro argomenti.

1. Scostamento orizzontale. Un qualunque valore positivo indica una che l'ombreggiatura apparirà alla destra dell'elemento; viceversa, ogni valore negativo posiziona l'ombreggiatura a sinistra dell'elemento stesso.

2. Scostamento verticale. Un qualunque valore positivo posiziona l'ombreggiatura più in basso rispetto all'elemento; viceversa, un valore negativo la colloca più in alto.
3. Lunghezza del raggio di sfocatura. Più alto è questo valore, più l'ombreggiatura sarà sfumata. Se posto pari a zero, l'ombreggiatura sarà netta e non presenterà alcuna sfumatura.
4. Colore. Il colore applicato all'effetto di ombreggiatura.

Per ottenere l'effetto illustrato in Figura 13.7 sarà sufficiente scrivere:

**LISTATO 13.6** Proprietà box-shadow: un esempio

```
-webkit-box-shadow: 8px 8px 3px #ccc;
-moz-box-shadow: 8px 8px 3px #ccc;
box-shadow: 8px 8px 3px #ccc;
```

Per garantirci un supporto adeguato utilizziamo anche i prefissi `-webkit-` e `-moz-` ancora necessari per Firefox, Chrome e Safari.

La specifica, riguardo alla proprietà `box-shadow`, offre una possibile applicazione che va oltre l'aspetto stilistico fine a se stesso, come può essere quello di un'ombreggiatura applicata a un blocco. Si può utilizzare questa proprietà per enfatizzare un termine in una frase.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

**FIGURA 13.8** Un suggerimento proposto dalla specifica per enfatizzare uno o più termini che occorrono in un testo.

Il listato che segue mostra la regola applicata per produrre un tale effetto.

**LISTATO 13.7** Un'applicazione pratica della proprietà box-shadow

```
em {
  font-style: normal;
  border: 1px solid #efefef;
  -webkit-box-shadow: 4px 4px 0px #ccc;
  -moz-box-shadow: 4px 4px 0px #ccc;
  box-shadow: 4px 4px 0px #ccc;
}
```

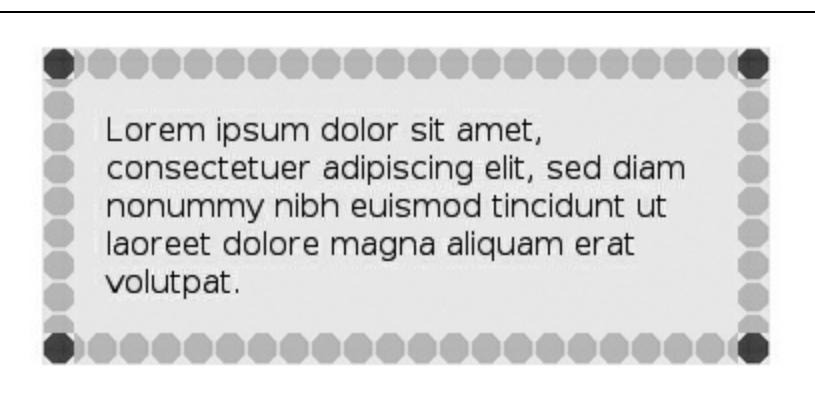
## Immagini applicate ai bordi: border-image

### TABELLA 13.3 Supporto della proprietà border-image

PROPRIETÀ	BROWSER
border-image	Chrome2+*, Firefox 3.5*, Opera10.5+ Safari 3+*

\* Proprietà implementata solo per mezzo dei prefissi proprietari `-webkit-` e `-moz-`.

Se finora abbiamo avuto solo la possibilità di impostare lo stile del bordo scegliendo uno tra gli 11 valori predefiniti per la proprietà `border-style`, oggi abbiamo anche la facoltà di aggiungere un'immagine che possa essere replicata. L'esempio, di dubbia qualità stilistica, ma utile a dare un'idea di quanto si possa ottenere, è illustrato in Figura 13.9.



**FIGURA 13.9** border-image permette di utilizzare un'immagine come bordo di un elemento.

L'immagine utilizzata per produrre questo bordo è ingrandita e illustrata di seguito.



**FIGURA 13.10** A sinistra l'immagine utilizzata per produrre la decorazione del bordo; a destra la stessa immagine divisa in sezioni.

Questa piccola immagine viene "spezzata" in nove parti distinte. Otto di queste nove parti, ossia tutte a eccezione della parte centrale, verranno utilizzate per riprodurre il bordo dell'elemento secondo quanto stabilito dalla regola CSS. Il Listato 13.8 illustra la regola che ha prodotto il risultato osservabile in Figura 13.10.

#### LISTATO 13.8 Proprietà border-image: un esempio

```
border-image: url(border-image.png) 20 round;
```

- `url(border-image.png)` imposta il percorso dell'immagine da utilizzare come bordo.
- <sup>20</sup> è la distanza calcolata a partire da ciascuna estremità dell'immagine verso il centro. Serve per stabilire a che altezza tagliare l'immagine per produrre una griglia come quella proposta in Figura 13.10. In base a questo valore si interviene sulla parte di immagine che sarà utilizzata per creare il bordo.
- `round`. Indica il modo in cui l'immagine deve essere impiegata per produrre il bordo. I valori possibili sono tre: `stretch` (l'immagine è estesa al punto da rivestire l'area del bordo), `repeat` (le dimensioni dell'immagine non sono alterate, essa viene ripetuta tante volte quanto è necessario per rivestire il bordo); `round`: simile a `repeat` ma con l'accortezza di un'immagine che si adatta alle dimensioni del bordo.

Abbiamo appreso come la proprietà `border-image` consenta di impostare diverse informazioni in modo sintetico. Si tratta infatti di una forma contratta. Le stesse informazioni possono essere impostate utilizzando un maggior numero di proprietà, ognuna delle quali svolge un compito ben preciso.

- `border-image-source` che accetta come valore il percorso dell'immagine che verrà applicata come bordo.
- `border-image-slice` utile a definire la forma della griglia che divide l'immagine in 9 parti e ne determina l'utilizzo ai fini della rappresentazione del bordo. Il valore può essere rappresentato in pixel o in percentuale.
- `border-image-repeat` che accetta uno dei tre valori possibili, ossia `stretch`, `repeat`, `round`. `stretch` è il valore predefinito, ove non ne venga esplicitamente impostato uno diverso.

Ne consegue che il Listato 13.8 si sarebbe potuto scrivere anche come proposto di seguito.

#### **LISTATO 13.9** Sintassi più verbosa per esprimere la stessa regola indicata nel Listato 13.8

---

```
border-image-source: url(border-image.png);
border-image-slice: 20;
border-image-repeat: round;
```

## Sfondi

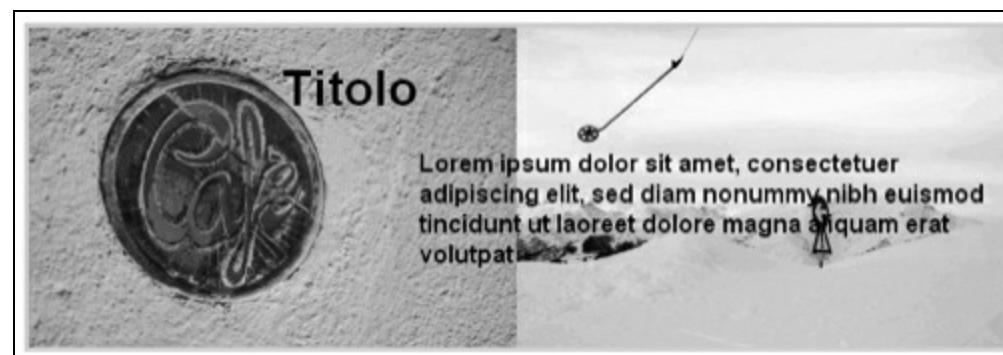
### Sfondi multipli

Il modulo CSS3 relativo a bordi e sfondi permette di raggiungere un controllo più raffinato

anche sulle immagini usate come sfondo. Per esempio, finora si è sempre stati limitati all'adozione di un'unica immagine di sfondo per elemento. Questo vincolo ora può essere superato. Il Listato 13.10 illustra come ottenere il risultato visibile in Figura 13.11, dove due immagini diverse sono impiegate come sfondo per uno stesso marcatore `<div>`.

### **LISTATO 13.10** Sfondi multipli applicati a un unico elemento

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>tit esempio</title>
<style>
body {font-family: arial, verdana, sans-serif; font-weight: bold;}
h1 {font-family: sans-serif; margin-left: 5.2em; }
p {margin-left: 16em; }
div {
width: 640px;
height: 210px;
border: 3px solid #ccc;
background: url(caffè.jpg) top left no-repeat,
url(neve.jpg) top right no-repeat
}
</style>
</head>
<body>
<div>
<h1>Titolo</h1>
<p>Lorem ipsum dolor [omissis]</p>
</div>
</body>
</html>
```



**FIGURA 13.11** Sfondi multipli: 2 immagini distinte come sfondo per uno stesso elemento.

## TABELLA 13.4 Supporto della proprietà background-size

PROPRIETÀ	BROWSER
background-size	Chrome 3+*, Firefox 3.6+*, IE9, Opera 9.5+*, Safari 5*

\* Proprietà supportata per mezzo dei prefissi proprietari: `-webkit-`, `-moz-` e `-o-`.

Le dimensioni delle immagini di sfondo hanno sempre rappresentato un aspetto non suscettibile di modifiche. Di ciò si è sempre dovuto tenere conto nella fase di ideazione del layout di un sito web. Grazie alla proprietà `background-size` questo non è più vero. È possibile impostare dimensioni diverse da quelle reali sia ricorrendo a valori assoluti, espressi per esempio in pixel, o relativi, per mezzo di valori percentuali. Il Listato 13.11 propone un esempio.

### LISTATO 13.11 Impostare dimensioni relative per l'immagine di sfondo

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>CSS3: dimensioni relative dell'immagine di sfondo</title>
    <style>
      body {font-family: arial, verdana, sans-serif; font-weight: bold;}
      h1 {font-family: sans-serif; margin-left: 10em; }
      p {margin: 1em 0.4em 1.2em 18em; }

      div {
        background: url(gin-bkg-size.jpg);
        -webkit-background-size: 36% 100%;
        -o-background-size: 36% 100%;
        -moz-background-size: 36% 100%;
        background-size: 36% 100%;
        background-repeat: no-repeat;
        padding: 60px 5px 5px 5px;
        border: 1px solid #900; color: #ccc"
      }
    </style>
  </head>
  <body>
    <div>
      <!-- segue breve testo -->
    </div>
  </body>
</html>
```

Nel Listato 13.11 la proprietà `background-size` è fatta precedere dalla stessa proprietà con i prefissi proprietari. In questa fase, tale precauzione risulta necessaria per disporre di un

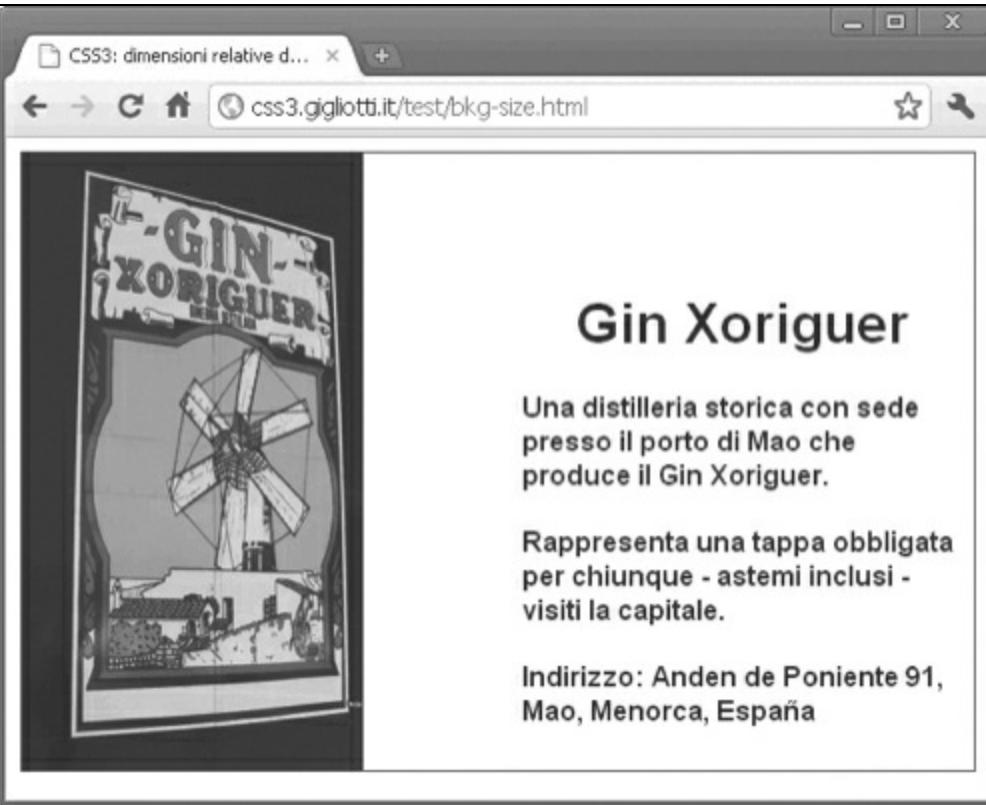
supporto più esteso. Avendo impostato dimensioni solo in termini percentuali, l'immagine di sfondo si adatterà a ogni modifica del `<div>` all'interno del quale opera: le Figure 13.12 e 13.13 ne sono una dimostrazione.



**FIGURA 13.12** Le dimensioni dell'immagine di sfondo sono impostate in relazione all'elemento contenitore `<div>`.

La proprietà `background-size` ammette quattro diverse modalità di definizione delle dimensioni.

- **Valori assoluti:** è possibile definire le dimensioni in valori assoluti utilizzando, per esempio, i pixel. In `background-size: 480px 568px;`, la prima misura indica la larghezza, la seconda esprime la lunghezza dell'immagine.
- **Valori relativi:** dimensioni relative all'elemento contenitore possono essere impostate per mezzo di valori percentuali. Con `background-size: 36% 100%;`, l'immagine di sfondo si estenderà in larghezza per circa un terzo delle dimensioni dell'elemento contenitore e ne occuperà, in lunghezza, l'intera estensione.



**FIGURA 13.13** L'immagine di sfondo altera le proprie dimensioni per mantenere invariate le percentuali di spazio occupato nell'elemento contenitore.

- **Contain:** applica all'immagine le dimensioni massime consentite dall'elemento contenitore avendo cura di preservare il rapporto d'aspetto, ossia il rapporto tra altezza e lunghezza, dunque evitando distorsioni come quelle apprezzabili in Figura 13.12 e 13.13.
- **Cover:** le dimensioni dell'immagine di sfondo sono tali da occupare per intero l'area dell'elemento contenitore. Il tutto avviene preservando il rapporto d'aspetto dell'immagine.

Il valore predefinito di questa proprietà è `auto`, che propone l'immagine nelle sue dimensioni originali.

#### NOTA

Nel caso in cui sia espressamente specificata solo una delle due dimensioni, sia in valore assoluto sia in valore percentuale, si assume che l'altra abbia valore `auto`. Ciò implica che verrà mantenuto inalterato il rapporto d'aspetto dell'immagine.

### background-origin e background-clip

Le proprietà `background-origin` e `background-clip` incidono sul posizionamento dell'immagine di sfondo in relazione all'elemento contenitore. La Tabella 13.5 illustra il supporto di queste proprietà.

**TABELLA 13.5** Supporto delle proprietà `background-origin` e `background-clip`

PROPRIETÀ	BROWSER
<code>background-origin</code>	Chrome3+*, Firefox 3.5*+, IE9**, Opera10.5+, Safari 5*
<code>background-clip</code>	Chrome3+*, Firefox 3.5*+, IE9**, Opera10.5+, Safari 5*

\* Firefox, Chrome e Safari implementano la proprietà nella versione con prefisso rispettivamente di `-moz-` e `-webkit-`. Firefox4 implementa la funzionalità senza ricorso a prefissi.

\*\* IE9 implementa la proprietà in modo non corretto.

La proprietà `background-origin` interviene sulla posizione dell'immagine di sfondo di un elemento per mezzo di uno dei tre valori che essa può assumere: `border-box`, `padding-box` e `content-box`:

- `border-box`: indica che il vertice posto in alto a sinistra dell'immagine coincide con il vertice sinistro esterno del bordo (Figura 13.14); ciò implica una sovrapposizione del bordo sull'immagine.



**FIGURA 13.14** Il vertice del bordo esterno sinistro dell'elemento contenitore corrisponde al vertice esterno sinistro dell'immagine di sfondo.

- `padding-box`: indica che il vertice posto in alto a sinistra dell'immagine coincide con il vertice sinistro esterno del `padding` (Figura 13.15). Questo è il valore predefinito nel caso in cui non ne sia espressamente indicato uno diverso.

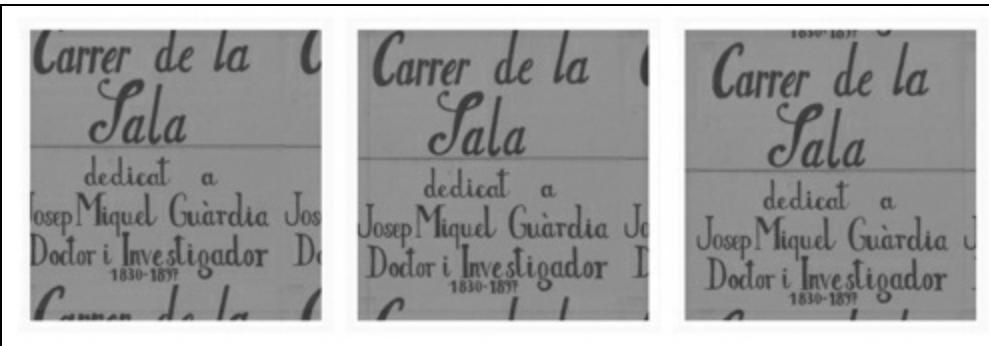


**FIGURA 13.15** Il vertice esterno sinistro dell'area del padding dell'elemento contenitore corrisponde al vertice esterno sinistro dell'immagine.

- `content-box`: indica che il vertice posto in alto a sinistra dell'immagine coincide con il vertice sinistro esterno dell'area del contenuto (Figura 13.16).



**FIGURA 13.16** Il vertice esterno destro dell'area del contenuto coincide con il vertice esterno sinistro dell'immagine.



**FIGURA 13.17** Effetto dei diversi valori per la proprietà: `background-origin`.

La Figura 13.17 illustra il risultato prodotto con il Listato 13.12. Da sinistra verso destra si osserva l'impatto del valore assegnato alla proprietà `background-origin` sull'immagine di sfondo, che sembra slittare sempre più in basso a destra man mano che si passa dal valore `border-box` (primo `<div>` a sinistra) al valore `content-box`.

La proprietà `background-clip` invece ha lo scopo di stabilire se l'immagine di sfondo si estenda o meno nell'area occupata dal bordo. La proprietà accetta due possibili valori: `border-box` e `padding-box`. Il primo dei due, che ne rappresenta anche il valore predefinito, stabilisce che l'immagine si estenda anche lungo l'area del bordo mentre se si esplicita il valore `padding-box` ne risulta un'immagine di sfondo che non si applica al bordo.

#### **LISTATO 13.12** Proprietà `background-origin`: un esempio

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
```

```

<title>background-origin</title>

<style>
div {
  width: 222px;
  height: 222px;
  border: 10px solid yellow;
  padding: 10px;
  background: url(bkg-origin.gif);
  float: left;
  margin: 5px;
}

#test1 {
  -moz-background-origin: border-box;
  -webkit-background-origin: border-box;
  background-origin: border-box;
}

#test2 {
  -moz-background-origin: padding-box;
  -webkit-background-origin: padding-box;
  background-origin: padding-box;
}

#test3 {
  -moz-background-origin: content-box;
  -webkit-background-origin: content-box;
  background-origin: content-box;
}

</style>
</head>
<body>
  <div id="test1"></div>
  <div id="test2"></div>
  <div id="test3"></div>
</body>
</html>

```

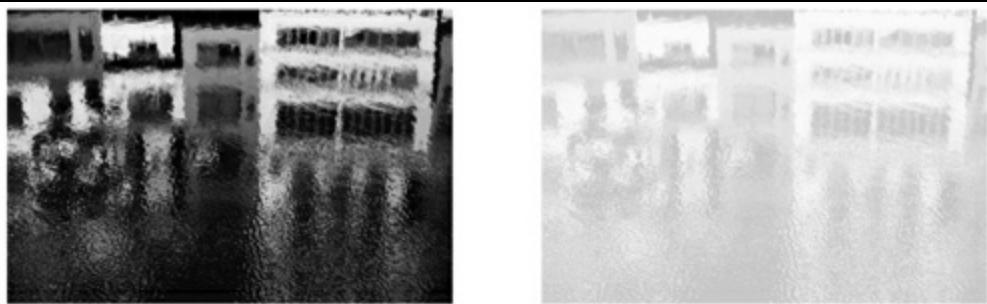
## Opacità

È possibile intervenire sul grado di trasparenza di un elemento mediante la proprietà `opacity` che accetta valori da `0` a `1` estremi compresi, dove `0` rappresenta un'immagine del tutto invisibile e `1` rappresenta elementi e contenuti in un contesto privo di trasparenza. La Tabella 13.6 ne rappresenta il supporto.

**TABELLA 13.6** Supporto della proprietà `opacity`

\* Versioni di Internet Explorer precedenti alla versione 9 offrono la medesima funzionalità per mezzo della proprietà `filter`.

In Figura 13.18 mettiamo a confronto testo e immagini senza trasparenza e, sulla destra, gli stessi elementi cui risulta applicato un grado di opacità solo pari allo 0.3.



**FIGURA 13.18** A sinistra un'immagine senza alcun filtro che incida sull'opacità. A destra, la stessa immagine e testo con opacità pari a 0.3.

### **LISTATO 13.13** Proprietà opacity: un esempio

```
#trasparente {  
    opacity: 0.3;  
    filter:alpha(opacity=30);  
}
```

## Riferimenti alle risorse citate e altri link utili

- La specifica ufficiale pubblicata dal W3C su bordi e sfondi: <http://www.w3.org/TR/css3-background>
- Una tabella di supporto straordinariamente dettagliata sulla proprietà `border-radius`: <http://muddledramblings.com/table-of-css3-border-radius-compliance>
- Questo articolo spiega come riprodurre i puntini di sospensione in Firefox: <http://www.jide.fr/english/emulate-text-overflowellipsis-in-firefox-with-css>

## Conclusioni

Il progetto tipografico su Web sta allineandosi per ricchezza di strumenti e raffinatezza a quanto esiste già da tempo in settori più tradizionali, come quello dell'editoria su carta. I vari moduli CSS di terzo livello, inoltre, pur presentando nel loro complesso un insieme disomogeneo di specifiche per via del diverso stadio di maturità in cui queste si trovano, rappresentano uno dei componenti fondamentali di quell'evoluzione del Web che con-

HTML5 sta dando tanta concretezza a un nuovo modo di concepire la propria presenza in Rete.

# Capitolo 14

# Trasformazioni

## Funzioni di trasformazione

Applicare una o più trasformazioni a un elemento HTML significa intervenire sulle coordinate che ne determinano la posizione.

Le trasformazioni hanno luogo applicando agli elementi che ne sono oggetto, come le immagini della Figura 14.1, coordinate diverse sul piano bidimensionale. A questo scopo si utilizza la proprietà `transform`. Tale proprietà altera la posizione di un dato elemento per mezzo di funzioni di trasformazione (`rotate`, `skew`, `scale` e `translate` sono le principali).

**TABELLA 14.1** Supporto delle proprietà relative al modulo delle trasformazioni

PROPRIETÀ	BROWSER
<code>transform*</code>	Chrome2+, Firefox 3.5+, IE9, Opera10.5+ Safari 3.1+
<code>transform-origin*</code>	Chrome5+, Firefox 3.5+, IE9, Opera10.5+ Safari 3.1+

\* L'implementazione dei principali browser è ancora fortemente legata ai rispettivi prefissi proprietari. Internet Explorer 9 è l'unico browser che riconosce le proprietà senza l'ausilio del prefisso.

Per modificare il posizionamento delle immagini come illustrato in Figura 14.1 sono state aggiunte le seguenti regole CSS al Listato 11.20.

**LISTATO 14.1** Trasformazioni

```
#mercado {  
    -ms-transform: rotate(10deg);  
    -moz-transform: rotate(10deg);  
    -webkit-transform: rotate(10deg);  
    -o-transform: rotate(10deg);  
    transform: rotate(10deg);  
}  
  
#cattedrale {  
    -ms-transform: skew(10deg);  
    -moz-transform: skew(10deg);  
    -webkit-transform: skew(10deg);  
}
```

```
-o-transform: skew(10deg);  
transform: skew(10deg);  
}  
  
#paella {  
-ms-transform: scale(0.8,0.8);  
-moz-transform: scale(0.8,0.8);  
-webkit-transform: scale(0.8,0.8);  
-o-transform: scale(0.8,0.8);  
transform: scale(0.8,0.8);  
}  
  
#museo {  
-ms-transform: translate(10px,20px);  
-moz-transform: translate(10px,20px);  
-webkit-transform: translate(10px,20px);  
-o-transform: translate(10px,20px);  
transform: translate(10px,20px);  
}
```

## NOTA

---

Il codice che si ottiene, purtroppo, è molto verboso per via di un'implementazione ancora dipendente da estensioni proprietarie: quei prefissi legati ai singoli browser che nel lungo periodo spariranno per via della graduale adozione delle proprietà ufficiali, ma attraverso i quali oggi bisogna passare se si vuole usufruire delle funzionalità illustrate.

Nel Listato 14.1 non abbiamo fatto altro che trarre vantaggio da alcune delle funzioni che possono essere assegnate alla proprietà `transform`. Ogni funzione sottintende una diversa tipologia di trasformazione; ne analizziamo di seguito le caratteristiche.

## rotate()

Ruota l'elemento in senso orario per il numero di gradi specificati come argomento. Valori negativi determinano una rotazione in senso antiorario. Un valore pari a 360 gradi lascia inalterato l'elemento, valori superiori sono consentiti e vengono convertiti in valori positivi compresi tra 0 e 360.

pseudo classe :nth-child - Mozilla Firefox  
 File Modifica Visualizza Gruologra Segnalibri Strumenti Aiuto  
 http://css3.gigliotti.it/test/transform.html Google

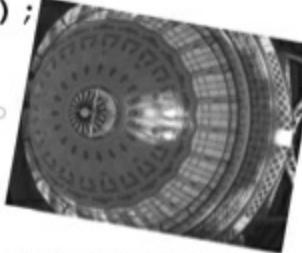
**Valencia in un giorno**

Scritto da Gabriele Gigliotti  
 Pubblicato il 07-11-2010.

**rotate(10deg);**  
 Abbiamo lasciato Bergamo una fredda mattina di tardo ottobre. Erano le 7.45. Saremmo ritornati tredici ore più tardi. Due ore dopo ci stavamo godendo i venti gradi con cui Valencia ci ha accolto sin da subito. Il cambio di temperatura è stato così eclatante e raggiunto in così poco tempo che abbiamo avuto la sensazione di fare un viaggio nel tempo più che nello spazio. Ci sembrava di essere tornati in primavera!



**skew(10deg);** Iati da un infallibile senso dell'orientamento (*non il mio, decisamente non il mio*) abbiamo raggiunto presto il centro della città e dove abbiamo visitato il Mercado Central (non prima di aver assaggiato una buona einsemada). Lì abbiamo scattato le prime foto e abbiamo comprato frutta e acqua per il resto della giornata. Lasciato il mercato abbiamo fatto tappa alla cattedrale



**scale(0.8, 0.8);**  
 Dalla sua torre "La Miguelete" si apprezza una stupenda vista della città. Nel frattempo era quasi diventata ora di pranzo, lo stomaco brontolava e il Riu era abbastanza vicino. Abbiamo aspettato parecchio prima di assaggiare una paella memorabile, ne è valsa la pena :-)



**translate(10px, 20px);**  
 Abbiamo trascorso il pomeriggio facendo una passeggiata lungo quello che un tempo era il letto del fiume Turia e che è stato convertito in un parco che si estende per chilometri (fossimo stati in Italia avrebbe avuto la meglio il piano alternativo di trasformare il letto del torrente in un'autostrada). Lungo questo torrente si trova la "Ciutat de les Arts i les Ciències" un complesso architettonico concepito da Calatrava che è diventato uno dei simboli della città



[\[Leggi la versione integrale\]](#). Per segnalare errori di stampa, materiale di approfondimento o suggerimenti per prossimi articoli scrivi a [gabriele.gigliotti@gmail.com](mailto:gabriele.gigliotti@gmail.com)

Completo

**FIGURA 14.1** Esempi di diversi tipi di trasformazione applicati alle immagini.

**LISTATO 14.2** Trasformazione con funzione rotate: un esempio

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>transform rotate-function</title>
    <style>
      body {font: 0.8em/1.3em arial, sans-serif;}
      figure, figcaption {display: block;}
      div {
        margin: 4em 3em 2em;
        padding: 3em 0.2em 2em;
      }
    </style>
  </head>
  <body>
    <h1>Valencia in un giorno</h1>
    <img alt="Aerial view of Valencia city center with a 10deg rotation effect." data-bbox="11 275 200 375"/>
    <img alt="Cathedral of Valencia with a 10deg skew effect." data-bbox="415 170 600 290"/>
    <img alt="Paella dish with a 0.8 scale effect." data-bbox="435 385 575 465"/>
    <img alt="Funicular tower with a 10px, 20px translate effect." data-bbox="25 455 200 545"/>
    <div>
      <img alt="Aerial view of Valencia city center with a 10deg rotation effect." data-bbox="11 275 200 375"/>
      <img alt="Cathedral of Valencia with a 10deg skew effect." data-bbox="415 170 600 290"/>
      <img alt="Paella dish with a 0.8 scale effect." data-bbox="435 385 575 465"/>
      <img alt="Funicular tower with a 10px, 20px translate effect." data-bbox="25 455 200 545"/>
    </div>
  </body>
</html>
```

```
width: 320px;  
border: thin solid black; display: float;  
}  
/* rotazione in senso orario di 10 gradi */  
#family {  
-ms-transform: rotate(10deg);  
-moz-transform: rotate(10deg);  
-webkit-transform: rotate(10deg);  
-o-transform: rotate(10deg);  
transform: rotate(10deg);  
}  
/* rotazione in senso antiorario di 15 gradi */  
#street {  
-ms-transform: rotate(-15deg);  
-moz-transform: rotate(-15deg);  
-webkit-transform: rotate(-15deg);  
-o-transform: rotate(-15deg);  
transform: rotate(-15deg);  
}  
/* rotazione in senso orario di 20 gradi */  
#load {  
-ms-transform: rotate(380deg);  
-moz-transform: rotate(380deg);  
-webkit-transform: rotate(380deg);  
-o-transform: rotate(380deg);  
transform: rotate(380deg);  
}  
</style>  
</head>  
<body>  
<div id="family">  
<figure>  
  
<figcaption>Una famiglia indiana incontrata a Mamallapuram</figcaption>  
</figure>  
</div>  
<div id="street">  
<figure>  
  
<figcaption>Lungo le strade di Chennai</figcaption>  
</figure>  
</div>  
<div id="load">  
<figure>  
  
<figcaption>Una gran quantità di caschi di banane trasportati con grande perizia</figcaption>  
</figure>  
</div>
```

```
width="320" height="214">
<figcaption>Un trasporto complicato</figcaption>
</figure>
</div>
</body>
</html>
```

Il Listato 14.2 e la seguente Figura 14.2 propongono un'applicazione pratica in cui tre elementi `<div>` sono visualizzati rispettivamente con:

- una rotazione 10 gradi in senso orario (valore positivo): `rotate(10deg)`;
- una rotazione di 15 gradi in senso antiorario (valore negativo): `rotate(-15deg)`;
- una rotazione di 20 gradi in senso orario (valore positivo maggiore di 360):  
`rotate(380deg)` .



**FIGURA 14.2** Rotazione di 10 gradi in senso orario, di 15 gradi in senso antiorario e di 20 gradi in senso orario.

## NOTA

La funzione `rotate()` come anche la funzione `skew()` accettano sia valori espressi in gradi sia in radianti.

## scale(), scalex() e scaley()

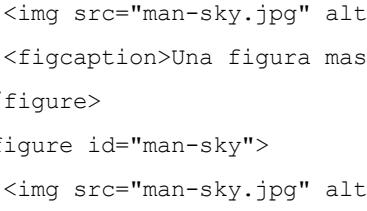
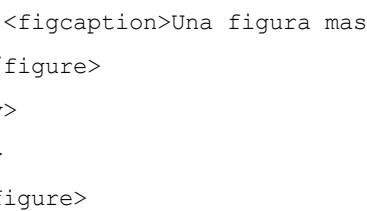
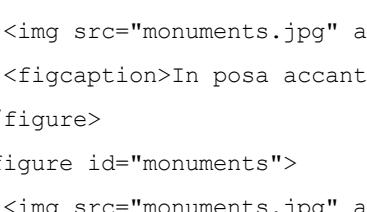
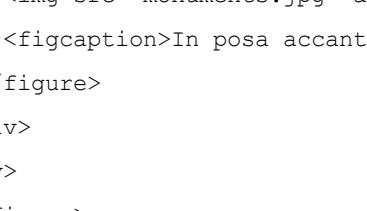
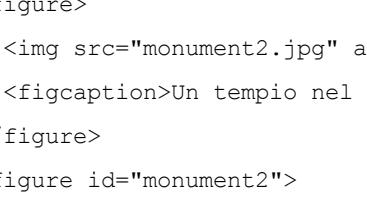
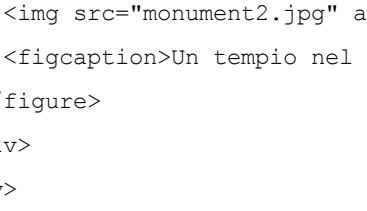
La funzione `scale()` modifica le dimensioni dell'immagine. Essa accetta uno o due argomenti: se il secondo non è specificato si assume sia identico al primo.

Il primo valore numerico esprime una variazione di larghezza dell'elemento; il secondo, se presente, imposta la variazione della lunghezza dell'elemento. I numeri sono espressi senza unità di misura.

### LISTATO 14.3 Funzione di trasformazione scale(): un esempio

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>transform scale function</title>
    <style>
      body {font: 1em/1.3em arial, sans-serif;}
      figure, figcaption {
        display: block;
        margin: 0em 3em 4em;
      }
      div {
        float:left;
      }
      #man-sky {
        -ms-transform: scale(0.7);
        -moz-transform: scale(0.7);
        -webkit-transform: scale(0.7);
        -o-transform: scale(0.7);
        transform: scale(0.7);
      }
      #monuments {
        -ms-transform: scale(1.3);
        -moz-transform: scale(1.3);
        -webkit-transform: scale(1.3);
        -o-transform: scale(1.3);
        transform: scale(1.3);
      }
      #monument2 {
        -ms-transform: scale(-0.6);
        -moz-transform: scale(-0.6);
      }
    </style>
  </head>
  <body>
    <div>
      <img alt="A small version of the Colosseum and a large version of the Colosseum, both rotated 45 degrees." data-bbox="100 100 800 500"/>
    </div>
  </body>
</html>
```

```
-webkit-transform: scale(-0.6);
-o-transform: scale(-0.6);
transform: scale(-0.6);
}

</style>
</head>
<body>
<div>
<figure>

<figcaption>Una figura maschile si staglia contro il cielo.</figcaption>
</figure>
<figure id="man-sky">

<figcaption>Una figura maschile si staglia contro il cielo.</figcaption>
</figure>
</div>
<div>
<figure>

<figcaption>In posa accanto all'elefante. </figcaption>
</figure>
<figure id="monuments">

<figcaption>In posa accanto all'elefante</figcaption>
</figure>
</div>
<div>
<figure>

<figcaption>Un tempio nel sito di Mamallapuram. </figcaption>
</figure>
<figure id="monument2">

<figcaption>Un tempio nel sito di Mamallapuram </figcaption>
</figure>
</div>
</body>
</html>
```



**FIGURA 14.3** In basso la versione “trasformata” delle foto e relative didascalie.

Nel primo caso si è optato per un valore positivo ma minore di 1. Questo produce l’effetto di una riduzione delle dimensioni dell’elemento (in questo caso dell’immagine e della relativa didascalia).

Impostare un valore positivo maggiore di 1 per la funzione `scale()` produce un aumento delle dimensioni dell’elemento.

Se si invoca questa funzione passando come argomento un valore negativo, l’elemento appare diminuito o ingrandito a seconda del valore assoluto, oltre a risultare capovolto. Nell’esempio abbiamo sempre utilizzato la funzione `scale()` impiegando un unico argomento. Come detto in apertura di paragrafo, questo significa che si assume il secondo argomento identico al primo: ciò permette di variare le dimensioni dell’elemento pur nel rispetto delle sue proporzioni, che rimangono invariate.

Con le funzioni `scalex()` e `scaley()`, invece, si introducono distorsioni nella proporzione tra le dimensioni dell’elemento. Ciò accade perché tramite la funzione `scalex()` si modifica solo la dimensione orizzontale, lasciando inalterata quella verticale rispetto all’originale.

Viceversa, con `scaley()` si interviene solo sulla lunghezza dell’elemento; la larghezza non subisce variazioni.

Se applichiamo queste trasformazioni a una foto l’alterazione delle proporzioni è evidente. Un esempio pratico applicato a due delle tre foto del precedente listato rende il tutto più evidente.

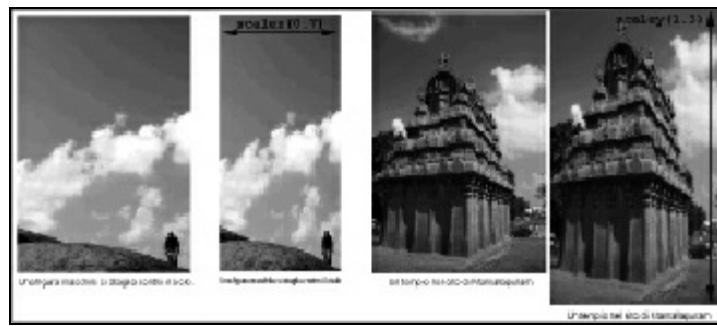
#### **LISTATO 14.4** `scalex()` `scaley()`: un esempio

```

-ms-transform: scalex(0.7);
-moz-transform: scalex(0.7);
-webkit-transform: scalex(0.7);
-o-transform: scalex(0.7);
transform: scalex(0.7);
}

#monument2 {
-ms-transform: scaley(1.3);
-moz-transform: scaley(1.3);
-webkit-transform: scaley(1.3);
-o-transform: scaley(1.3);
transform: scaley(1.3);
}

```



**FIGURA 14.4** scalex() altera solo la larghezza dell'elemento mentre scaley() agisce solo sulla lunghezza dello stesso.

## skew(), skewx() e skewy()

Queste funzioni di trasformazione producono l'inclinazione degli elementi cui sono applicate. L'effetto che si ottiene è quello di un rettangolo che, tirato verso l'esterno da due angoli opposti, si allunga fino ad assomigliare a un rombo.

In modo analogo a quanto abbiamo visto per la funzione `scale()`, qui esiste una funzione `skew()` che accetta due argomenti, di cui il secondo opzionale: se assente, non si applicherà alcuna trasformazione lungo l'asse verticale. Gli argomenti devono rappresentare gradi (o radianti) in base ai quali la trasformazione sarà applicata rispettivamente lungo l'asse orizzontale (primo argomento) e quello verticale (secondo argomento).

Per quanto sin qui detto ne consegue che le due dichiarazioni riportate nel Listato 14.5 sono equivalenti.

**LISTATO 14.5** L'inclinazione di un asse può definirsi in gradi o radianti

```

transform: skew(10deg);
transform: skew(0.17rad);

```

I 10 gradi della prima dichiarazione equivalgono a 0.17 radianti della seconda

## NOTA

Qui è stato applicato un certo grado di approssimazione poiché il valore corretto è, in realtà, 0,174532925. La regola per ottenere il valore radiante corrispondente è stata indicata nel Capitolo 6 a proposito dell'elemento `<canvas>`.

Di seguito un esempio relativo alla funzione `skew()`.

### **LISTATO 14.6** Funzione di trasformazione `skew()`: un esempio

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>transform skew function</title>
    <style>
      body {margin: 5em;}
      figcaption {display: block;}
      div {
        border: thin dotted black;
        padding: 3px;
        width: 300px;
        -ms-transform: skew(10deg,20deg);
        -moz-transform: skew(10deg,20deg);
        -webkit-transform: skew(10deg,20deg);
        -o-transform: skew(10deg,20deg);
        transform: skew(10deg,20deg);
      }
    </style>
  </head>
  <body>
    <div>
      <figure>
        
        <figcaption>Il traffico, per le strade di Chennai, è spesso congestionato e la moto è fondamentale per riuscire a muoversi per la città in un tempo ragionevole.</figcaption>
      </figure>
    </div>
  </body>
</html>
```



**FIGURA 14.5** La distorsione provocata su entrambi gli assi (orizzontale e verticale) dalla funzione `skew()` è piuttosto evidente.

Le funzioni `skewX()` e `skewY()`, come a questo punto non avrete faticato a intuire, applicano una distorsione solo a uno dei due assi di quel sistema di coordinate lungo il quale è disposto ciascun elemento di un documento HTML. L'esempio che segue mostra come viene rappresentato uno stesso blocco `<div>` quando a esso è applicata la funzione `skewX(15deg)`, `<div>` di sinistra, e `skewY(15deg)`, `<div>` di destra.

#### **LISTATO 14.7** Funzioni d'itrasformazione `skewX()` e `skewY()`: un esempio

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>transform skewX() e skewY()</title>
```

```

<style>
div {
  float: left;
  width: 300px;
  border: thin dotted black;
  margin: 2em 5em;
  padding: 3px;
}
#test1 {
  -ms-transform: skewX(15deg);
  -moz-transform: skewX(15deg);
  -moz-transform: skewX(15deg);
  -webkit-transform: skewX(15deg);
  -o-transform: skewX(15deg);
  transform: skewX(15deg);
}
#test2 {
  -ms-transform: skewY(15deg);
  -moz-transform: skewY(15deg);
  -webkit-transform: skewY(15deg);
  -o-transform: skewY(15deg);
  transform: skewY(15deg);
}
</style>
</head>
<body>
<div id="test1">
  <figure>
    
    <figcaption><!-- testo descrittivo --></figcaption>
  </figure>
</div>
<div id="test2">
  <!-- il contenuto di questo elemento è identico al precedente -->
</div>
</body>
</html>

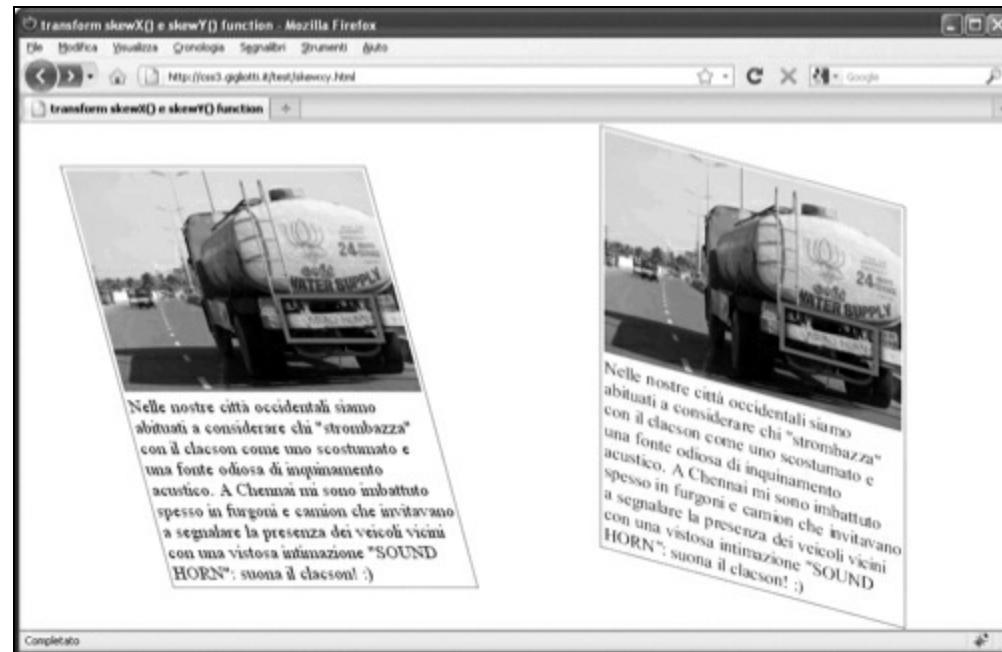
```

## translate(), translateX() e translateY()

Le tre funzioni di questo paragrafo sono relative a trasformazioni che consentono di spostare l'elemento cui si riferiscono di un numero di pixel definito attraverso i rispettivi argomenti lungo l'asse orizzontale e verticale.

translate() accetta uno o due argomenti. Il primo muove l'elemento lungo l'asse

orizzontale e verso destra. Il secondo, se specificato, sposta l'elemento verso il basso.



**FIGURA 14.6** Funzioni di trasformazione skewX() e skewY() applicate a un caso concreto.

`translateX()` e `translateY()` agiscono, rispettivamente, solo sull'asse orizzontale e solo sull'asse verticale.

#### **LISTATO 14.8** Funzione di trasformazione `translate()`: un esempio

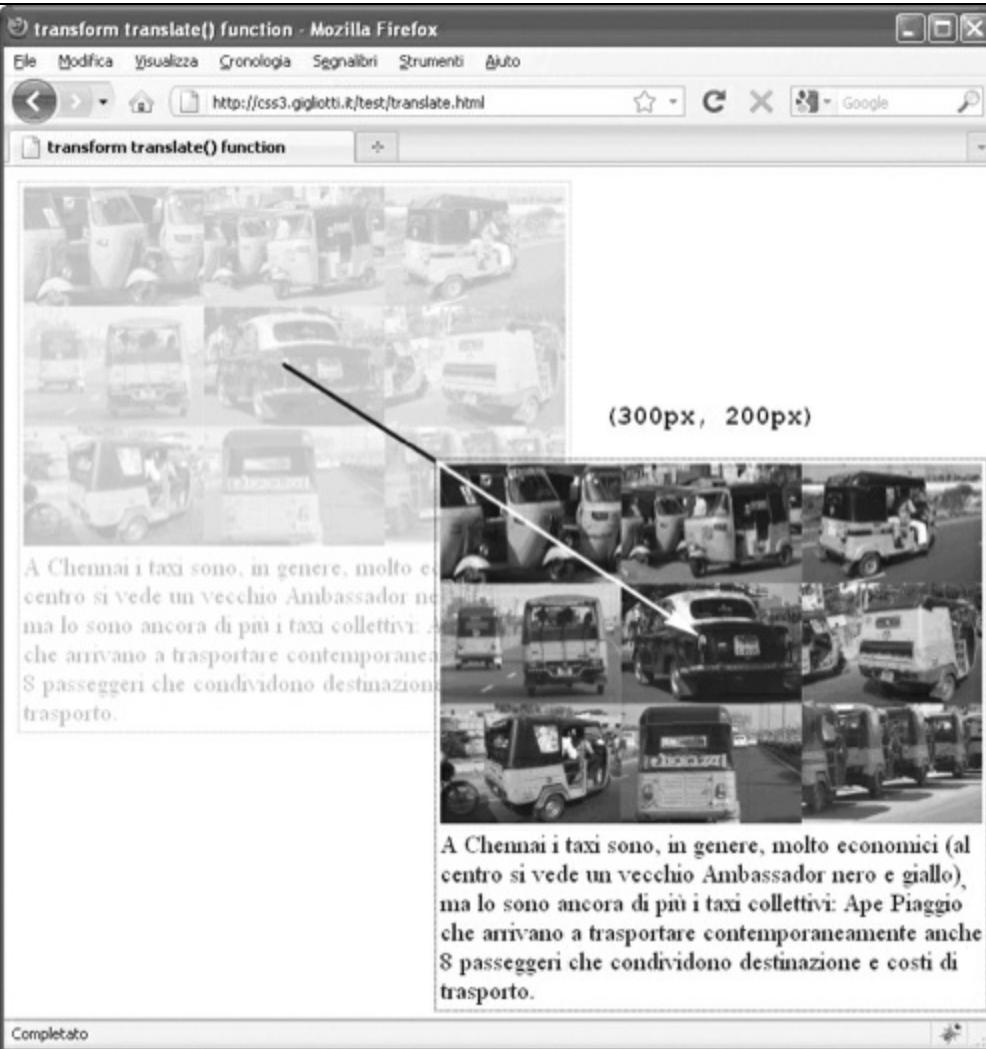
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>transform transform() function</title>
    <style>
      body {margin: 5em;}
      figcaption {display: block;}
      div {
        position: absolute;
        top: 50px;
        left: 30px;
        width: 390px;
        border: thin dotted black;
        padding: 3px;
      }
      #test1 {
        opacity: 0.3;
      }
      #test2 {
```

```

-ms-transform: translate(300px,200px);
-moz-transform: translate(300px,200px);
-webkit-transform: translate(300px,200px);
-o-transform: translate(300px,200px);
transform: translate(300px,200px);
}

</style>
</head>
<body>
<div id="test1">
<figure>
![Un collage di 9 foto con diversi tipi di taxi](taxi-please.jpg)
<figcaption><!-- testo descrittivo --><figcaption>
</figure>
</div>
<div id="test2">
<!-- il contenuto di questo elemento è identico al precedente -->
</div>
</body>
</html>

```



**FIGURA 14.7** Con la funzione translate() l'elemento può essere spostato in basso e a sinistra.

## transform-origin

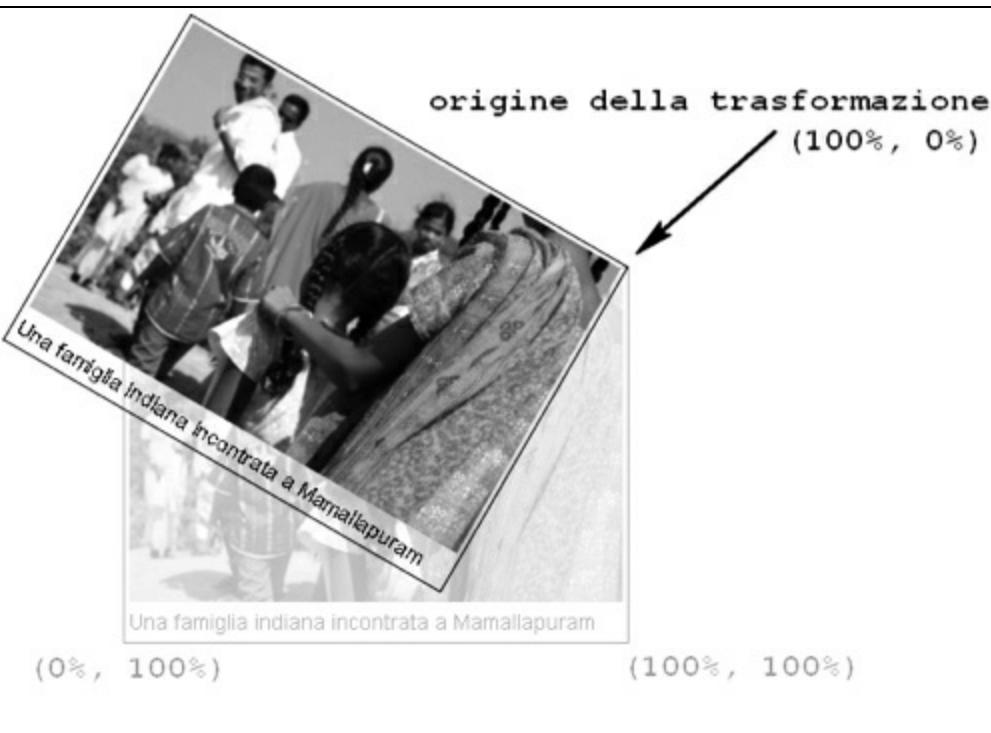
Tutte le trasformazioni di cui si è detto finora hanno origine dal centro dell'elemento, e questo perché `transform-origin`, la proprietà che abbiamo utilizzato implicitamente sfruttandone il valore predefinito `(50%, 50%)`, identifica proprio il centro dell'elemento. Volendo modificare un tale comportamento non dobbiamo far altro che variare i valori predefiniti. `transform-origin` accetta uno o due argomenti. Il primo dei due valori si riferisce al piano orizzontale, il secondo, se presente, incide sul piano verticale. Se non fosse presente sarebbe utilizzato il valore predefinito 50%.

Per modificare i valori predefiniti della proprietà `transform-origin` possiamo adottare un valore percentuale, un'unità di misura espressa in termini assoluti, oppure scegliere tra una delle parole chiave disponibili.

### **LISTATO 14.9** Attribuzione di valori percentuali alla proprietà `transform-origin`

```
-ms-transform-origin: 100% 0%;  
-moz-transform-origin: 100% 0%;  
-webkit-transform-origin: 100% 0%;  
-o-transform-origin: 100% 0%;  
transform-origin: 100% 0%;
```

Il Listato 14.9 identifica come origine della trasformazione l'angolo posto in alto a destra dell'elemento. In questo caso, se volessimo applicare una rotazione all'elemento, essa non avverrebbe più facendo perno attorno al centro dell'elemento ma attorno al vertice posto in alto a destra (Figura 14.8).



**FIGURA 14.8** Punto di origine della trasformazione definito per mezzo di valori percentuali (100%, 0%: angolo in alto a destra).

**LISTATO 14.10** Attribuzione di valori assoluti alla proprietà transform-origin

```
-ms-transform-origin: 0em 0em;  
-moz-transform-origin: 0em 0em;  
-webkit-transform-origin: 0em 0em;  
-o-transform-origin: 0em 0em;  
transform-origin: 0em 0em;
```

Il Listato 14.10 identifica il vertice posto in alto a sinistra. Ogni funzione di trasformazione avrà questo punto come riferimento per le modifiche da applicare (Figura 14.9).

## origine della trasformazione

(0em, 0em)

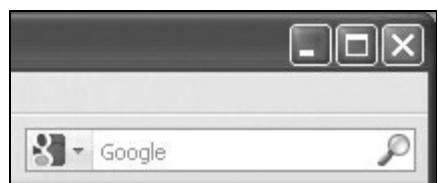


**FIGURA 14.9** Punto di origine della trasformazione definito per mezzo di unità di misura in valori assoluti (0em 0em: angolo in alto a sinistra).

### **LISTATO 14.11** Proprietà transform-origin: valori assegnati per mezzo di parole chiave

```
-ms-transform-origin: bottom right;  
-moz-transform-origin: bottom right;  
-webkit-transform-origin: bottom right;  
-o-transform-origin: bottom right;  
transform-origin: bottom right;
```

In tal caso è l'angolo posto in basso a destra che rappresenta l'origine della trasformazione, come illustrato in Figura 14.10.



**FIGURA 14.10** Punto di origine della trasformazione definito per mezzo di parole chiave (bottom left: angolo in basso a destra).

## Trasformazioni multiple

In tutto il capitolo abbiamo preso in considerazione esclusivamente l'elaborazione di

singole trasformazioni, tuttavia è possibile applicare in sequenza una serie di trasformazioni in modo da ottenere un risultato finale dato dalla somma degli effetti prodotti dalle singole funzioni di trasformazione. Potremo dunque scrivere:

#### **LISTATO 14.12** Applicazione concatenata di trasformazioni

```
transform: rotate(20deg) translate(100px, 50px) scale(1.2, 1.2);
```

In questo caso è stato sufficiente indicare una sequenza di tre funzioni di trasformazione separate da spazio perché l'elemento sia ruotato in senso orario di 20 gradi (rotate(20deg)), spostato di 100 pixel a destra e 50 pixel in basso (translate(100px, 50px)) e infine si veda applicato un ridimensionamento per il fattore 1.2 (scale(1.2, 1.2)). Poiché non specificato diversamente per mezzo della proprietà transform-origin, tutte queste funzioni troveranno nel centro dell'elemento l'origine di ogni processo di trasformazione.

## Riferimenti alle risorse citate e altri link utili

- Modulo CSS di terzo livello relativo alle trasformazioni: <http://www.w3.org/TR/css3-2d-transforms/>
- Documentazione della proprietà -moz-transform pubblicata sul Mozilla Developer Network: <https://developer.mozilla.org/En/CSS/-moz-transform>
- Documentazione della proprietà -moz-transform-origin pubblicata sul Mozilla Developer Network: <https://developer.mozilla.org/En/CSS/-moz-transform-origin>

## Conclusioni

Le trasformazioni possono rappresentare un ulteriore strumento a disposizione degli autori di pagine web per la realizzazione di effetti visivi accattivanti eppure semplici da ottenere; non c'è alcun componente aggiuntivo da installare, né altre pratiche esoteriche da portare a termine: solo (stra)ordinarie dichiarazioni CSS. Al momento, l'unico vero tallone d'Achille resta il supporto, limitato solo alle ultime versioni dei browser più diffusi. D'altro canto, se il loro impiego è ben studiato possono essere utilizzate per offrire di più a quegli utenti che hanno avuto la libertà, o più semplicemente la passione, di aggiornare il proprio browser senza diventare un impedimento alla fruizione dei contenuti per tutti gli altri.

# Capitolo 15

# Transizioni e animazioni

In questo capitolo trattiamo uno degli argomenti cui probabilmente è stato dato maggiore risalto rispetto a tutte le altre funzionalità introdotte con i moduli CSS di terzo livello: le transizioni e le animazioni ottenute esclusivamente per mezzo dei fogli di stile. Il motivo è evidente: attraverso un linguaggio dichiarativo riusciamo a ottenere effetti grafici di grande impatto. Finora questi risultati sono stati ottenuti solo al costo di un'intensa attività del linguaggio di scripting. Per un certo periodo si dovrà continuare a fare affidamento su JavaScript come modalità alternativa, se si ha intenzione di garantire lo stesso gradevole effetto di passaggio da uno stato iniziale a uno finale.

## Transizioni

**TABELLA 15.1** Supporto delle proprietà relative al modulo delle transizioni

PROPRIETÀ	BROWSER
transition-property*	Chrome3+, Firefox 4, Opera10.5+ Safari 3.1+
transition-duration*	Chrome3+, Firefox 4, Opera10.5+ Safari 3.1+
transition-timing-function*	Chrome3+, Firefox 4, Opera10.5+ Safari 3.1+
transition-delay*	Chrome3+, Firefox 4, Opera10.5+ Safari 3.1+

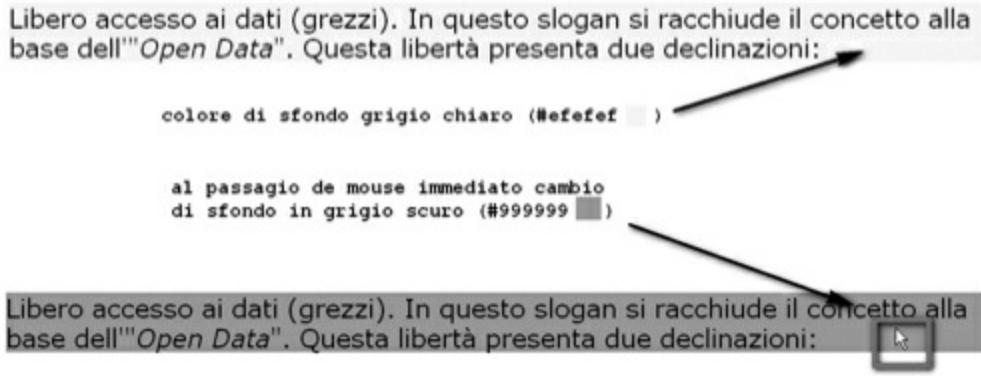
\* Il supporto è garantito solo per mezzo delle estensioni proprietarie.

Laddove interviene una modifica nel valore attribuito a una data proprietà CSS l'aggiornamento del valore produce, dal punto di vista visivo, un cambiamento pressoché istantaneo.

**LISTATO 15.1** Modifica immediata del valore di una proprietà

```
p {background-color: #efefef;}  
p:hover {background-color: #bbbbbb;}
```

Il Listato 15.1 permette di realizzare il cosiddetto effetto roll-over: al passaggio del puntatore del mouse il colore di sfondo del paragrafo cambia immediatamente da grigio chiaro (#efefef) a grigio scuro (#bbbbbb) per poi riacquisire con altrettanta rapidità il suo sfondo originario non appena il puntatore del mouse abbandona quel paragrafo.



**FIGURA 15.1** Il cambio di colore di sfondo al passaggio del mouse è istantaneo.

Una transizione CSS, per contro, descrive come il valore assegnato a una determinata proprietà possa cambiare gradualmente da un valore iniziale a uno finale in un dato arco temporale.

#### NOTA

Un effetto collaterale di questa tecnica è che se si interroga il valore della proprietà oggetto della transizione dal valore iniziale a quello finale, si potrà leggerne il valore intermedio assunto al momento della lettura.

Volendo schematizzare le componenti essenziali di un semplice processo di transizione, identifichiamo:

- in primo luogo la proprietà su cui applicare la transizione;
- è necessario poi definire per tale proprietà almeno due valori: quello corrente (il vecchio valore) e quello che verrà applicato in futuro (il nuovo valore);
- un evento, quello al determinarsi del quale ha inizio il processo di transizione;
- la durata della transizione, ossia il tempo che deve intercorrere affinché il vecchio valore sia gradualmente sostituito con quello nuovo.

Applichiamo ora questi concetti astratti a un caso concreto. Riprendiamo il testo dell'articolo usato come esempio nel Capitolo 2 e aggiungiamo le seguenti regole di stile al paragrafo.

#### LISTATO 15.2 Transizione applicata a un paragrafo

```

p {
    /* colore di sfondo iniziale */
    background-color: #efefef;
    /* proprietà interpretate solo da Firefox */
    -moz-transition-property: background-color;
    -moz-transition-duration: 3s;
    /* proprietà interpretate solo da Chrome e Safari */
    -webkit-transition-property: background-color;
    -webkit-transition-duration: 3s;
    /* proprietà interpretate solo da Opera */
    -o-transition-property: background-color;
    -o-transition-duration: 3s;
    /* Le proprietà ufficiali come definite nella specifica.
       È buona norma inserirle sempre, inoltre si avrà
       cura di aggiungerle solo dopo l'elenco delle proprietà
       con prefissi proprietari */
    transition-property: background-color;
    transition-duration: 3s;
}
p:hover {
    background-color: #999999;
}

```

## SUGGERIMENTO

---

Il colore `#999999` non è esattamente un colore adeguato come sfondo per un testo nero. È stato utilizzato solo per il suo elevato contrasto rispetto al valore iniziale, dunque idoneo, dal punto di vista didattico, a enfatizzare al meglio il concetto di transizione.

`transition-property` permette di impostare il nome della proprietà su cui sarà applicato l'effetto di transizione. In questo caso si tratterà del colore di sfondo del paragrafo: `background-color`. Il ricorso ai prefissi `-moz-` (per Firefox), `-webkit-` (per Chrome e Safari) e `-o-` (per Opera) si rende necessario in questa fase di implementazione del modulo relativo alle transizioni.

La dichiarazione `transition-duration: 3s;` ha lo scopo di definire l'intervallo temporale entro il quale si completerà il passaggio dal vecchio al nuovo valore: in questo caso stabiliamo che il tutto debba avvenire in 3 secondi.

Abbiamo dunque identificato oggetto (il colore di sfondo del paragrafo) e durata della transizione (3 secondi). La regola `p:hover` stabilisce che al passaggio del mouse sul paragrafo (evento) il colore di sfondo (oggetto) cambierà da grigio chiaro (`#efefef` vecchio valore) a grigio scuro (`#999999` nuovo valore). Poiché questa è la proprietà oggetto di transizione, verrà applicato un intervallo di tempo di 3 secondi (durata).

```
momento iniziale (sfondo #fefefef )
```

Libero accesso ai dati (grezzi). In questo slogan si racchiude il concetto alla base dell'"Open Data". Questa libertà presenta due declinazioni:

```
dopo 1 secondo (sfondo #cecece )
```

Libero accesso ai dati (grezzi). In questo slogan si racchiude il concetto alla base dell'"Open Data". Questa libertà presenta due declinazioni: 

```
dopo 2 secondi (sfondo #b9b9b9 )
```

Libero accesso ai dati (grezzi). In questo slogan si racchiude il concetto alla base dell'"Open Data". Questa libertà presenta due declinazioni: 

```
momento finale (sfondo #999999 )
```

Libero accesso ai dati (grezzi). In questo slogan si racchiude il concetto alla base dell'"Open Data". Questa libertà presenta due declinazioni: 

**FIGURA 15.2** Catturando alcuni momenti intermedi del processo di transizione diventa più evidente il passaggio graduale dal vecchio al nuovo valore.

## Transizioni multiple

Proviamo ora a simulare uno scenario appena più complicato per illustrare la duttilità delle proprietà alla base dell'effetto di transizione sin qui illustrato. Nel Listato 15.3 abbiamo ipotizzato che l'oggetto della transizione sia unico (nel nostro caso era appunto il colore di sfondo definito con `background-color`), tuttavia entrambe le proprietà `transition-property` e `transition-duration` possono accettare un elenco di valori separati da virgola.

### **LISTATO 15.3** Transizioni multiple

```
p {  
border: thin dotted white;  
background-color: #fefefef;  
-moz-transition-property: background-color, border;  
-moz-transition-duration: 3s, 4s;  
-webkit-transition-property: background-color, border;  
-webkit-transition-duration: 3s, 4s;  
-o-transition-property: background-color, border;  
-o-transition-duration: 3s, 4s;  
transition-property: background-color, border;  
transition-duration: 3s, 4s;  
}  
  
p:hover {  
border: thin dotted black;  
background-color: #999999;  
}
```

In questo caso sono due le proprietà oggetto di transizione: la prima (`background-color`)

è soggetta a una transizione tra valore iniziale e valore finale della durata di 3 secondi; per la seconda (`border`) l'intervallo temporale è diverso: 4 secondi.

## Modalità di transizione

Osservando attentamente la transizione, si potrebbe notare che il passaggio dal vecchio al nuovo valore non avviene lungo un percorso lineare ma sembra rallentare gradualmente tanto più si raggiunge il valore finale.

La modalità di trasformazione nell'intervallo di tempo dato è anch'essa modificabile grazie alla proprietà `transition-timing-function`. Essa ammette cinque diversi valori, che descriviamo di seguito.

- `ease`: tanto più ci si avvicina al valore finale, tanto più il processo di transizione rallenta. Esso rappresenta il comportamento predefinito. Nei due listati precedenti non è stato espressamente impostato un valore diverso, quindi la transizione è avvenuta secondo questo modello. Per avere un'idea più precisa del modello di transizione pensate a quei cassettoni della cucina che quando vengono chiusi, poco prima di arrivare a fondo corsa e modificare definitivamente la fisionomia della vostra mano rimasta inavvertitamente sospesa lungo la chiusura, rallentano in modo vistoso: ecco, questo è l'effetto di transizione `ease`.
- `linear`: progressione a velocità costante durante tutto l'arco temporale.
- `ease-in`: partenza veloce.
- `ease-out`: arrivo lento.
- `ease-in-out`: la somma dei due precedenti: inizia con brio e finisce lentamente.

### Per i più impavidi

In realtà le opzioni possibili sono 6 e non 5. La sesta opzione è:

`cubic-bezier(x1, y1, x2, y2)`.

Se tra le precedenti transizioni non siete riusciti a trovare l'effetto che fa per voi, la vostra unica possibilità è utilizzare questa opzione, in cui avrete cura di specificare dei valori opportuni per i 4 argomenti richiesti. Si tratta di definire una curva di Bézier (Figura 15.3) in cui i primi due argomenti (`x1` e `y1`) rappresentano le coordinate del punto P1 mentre la seconda coppia di argomenti (`x2` e `y2`) rappresenta le coordinate del punto P2. Il punto P0 avrà sempre coordinate `(0,0)`. Il punto P3 avrà sempre coordinate `(1,1)`. I punti P1 e P2 non potranno che assumere valori nell'intervallo da `0.0` a `1.0`, estremi inclusi.

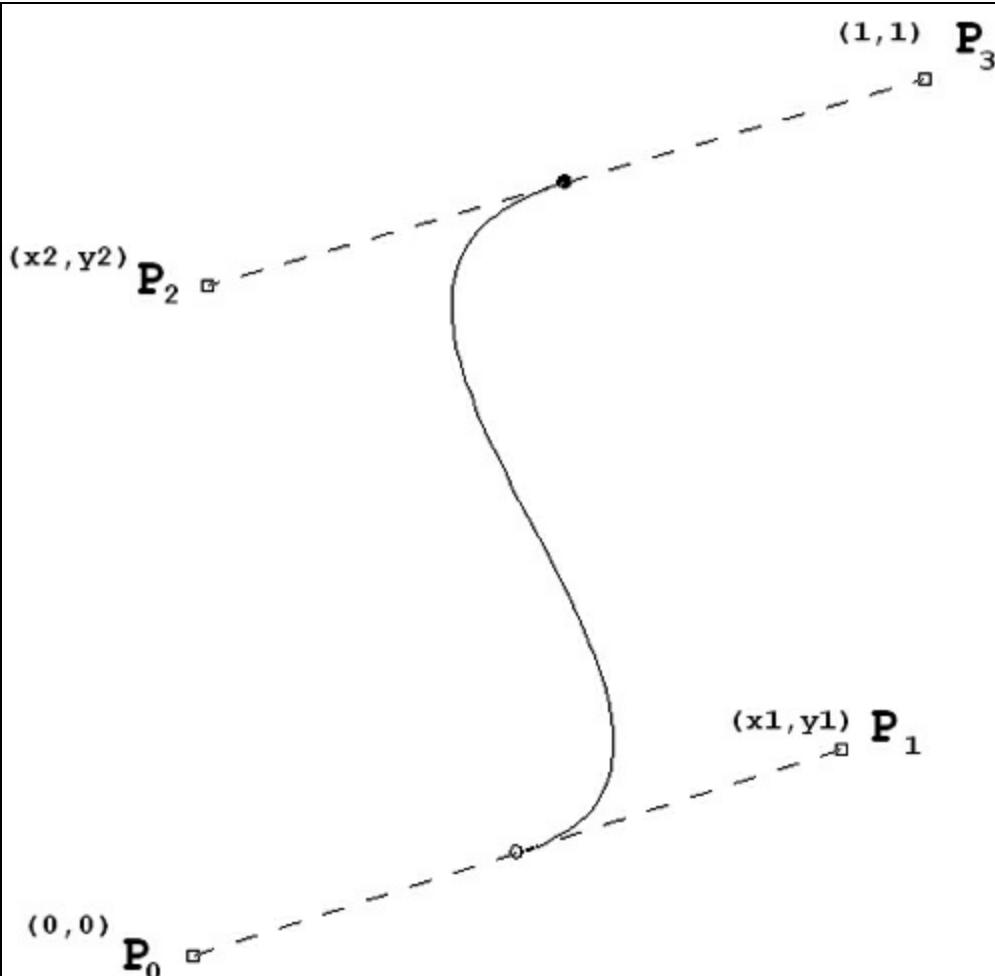
In effetti, le opzioni precedenti altro non sono che semplici "scorciatoie" per specifiche curve di Bézier ben definite.

**TABELLA 15.2** Costanti predefinite e relative curve di Bézier

**COSTANTE**

**CURVA DI BÉZIER**

ease	cubic-bezier(0.25, 0.1, 0.25, 1.0)
linear	cubic-bezier(0.0, 0.0, 1.0, 1.0)
ease-in	cubic-bezier(0.42, 0, 1.0, 1.0)
ease-out	cubic-bezier(0, 0, 0.58, 1.0)
ease-in-out	cubic-bezier(0.42, 0, 0.58, 1.0)



**FIGURA 15.3** La curva di Bézier è utilizzata per calcolare la modalità del processo di transizione.

Tornando sul Listato 15.3 per arricchirlo di questo ulteriore elemento, potremo scrivere quanto segue:

**LISTATO 15.4** Transizioni multiple lineari ed ease-in

```
p {
  border: thin dotted white;
  background-color: #efefef;
  -moz-transition-property: background-color, border;
  -moz-transition-duration: 3s;
  -moz-transition-timing-function: linear, ease-in;
  -webkit-transition-property: background-color, border;
  -webkit-transition-duration: 3s;
```

```

-webkit-transition-timing-function: linear, ease-in;
-o-transition-property: background-color, border;
-o-transition-duration: 3s;
-o-transition-timing-function: linear, ease-in;
transition-property: background-color, border;
transition-duration: 3s, 4s;
transition-timing-function: linear, ease-in;
}

p:hover {
  border: thin dotted black;
  background-color: #999999;
}

```

Stiamo così impostando due diverse transizioni. La prima riguarda il colore di sfondo, dura 3 secondi e ha luogo in forma lineare. La seconda riguarda il bordo, si completa in 4 secondi ed è di tipo `ease-in`.

Nelle transizioni che abbiamo realizzato il processo ha inizio subito dopo il verificarsi di un dato evento: nel nostro esempio si è trattato del passaggio del puntatore del mouse sul paragrafo. È possibile introdurre un ritardo tra il momento in cui ha luogo l'evento al quale è legata la transizione e il momento in cui essa ha inizio. Si realizza questo disallineamento per mezzo della proprietà `transition-delay`.

#### **LISTATO 15.5** Transizioni multiple con disallineamento tra evento e inizio effetto transizione

```

p {
  border: thin dotted white;
  background-color: #efefef;
  -moz-transition-property: background-color, border;
  -moz-transition-duration: 3s, 4s;
  -moz-transition-timing-function: linear, ease-in;
  -moz-transition-delay: 1s, 2s;
  -webkit-transition-property: background-color, border;
  -webkit-transition-duration: 3s, 4s;
  -webkit-transition-timing-function: linear, ease-in;
  -webkit-transition-delay: 1s, 2s;
  -o-transition-property: background-color, border;
  -o-transition-duration: 3s, 4s;
  -o-transition-timing-function: linear, ease-in;
  -o-transition-delay: 1s, 2s;
  transition-property: background-color, border;
  transition-duration: 3s, 4s;
  transition-timing-function: linear, ease-in;
  transition-delay: 1s, 2s;
}

```

```
p:hover {  
    border: thin dotted black;  
    background-color: #999999;  
}
```

Negli esempi sin qui proposti abbiamo sempre utilizzato diverse proprietà, ognuna delle quali svolge un compito ben determinato. È però possibile specificare tutti questi valori facendo uso di una sola proprietà dal nome `transition`, avendo cura di inserire i valori nel seguente ordine:

- la proprietà oggetto di modifica (ossia il valore di `transition-property`);
- la durata della transizione (ossia il valore di `transition-duration`);
- la modalità di esecuzione della transizione (ossia il valore di `transition-timing-function`);
- il periodo di tempo che intercorre tra lo scatenarsi dell'evento e l'inizio della transizione (ossia il valore di `transition-delay`).

Riscrivendo il Listato 15.5 con questa notazione otteniamo quanto segue.

#### **LISTATO 15.6** Il Listato 15.5 è stato riscritto ricorrendo alla sola proprietà `transition`

```
p {  
    border: thin dotted white;  
    background-color: #efefef;  
    -moz-transition: background-color 3s linear 1s, border 4s ease-in 2s;  
    -webkit-transition: background-color 3s linear 1s, border 4s ease-in 2s;  
    -o-transition: background-color 3s linear 1s, border 4s ease-in 2s;  
    transition: background-color 3s linear 1s, border 4s ease-in 2s;  
}  
  
p:hover {  
    border: thin dotted black;  
    background-color: #999999;  
}
```

#### nota

Nelle transizioni abbiamo sempre utilizzato un numero intero di secondi, ma avremmo potuto impiegare anche frazioni di secondi. Per esempio, per definire una transizione di mezzo secondo possiamo scrivere `0.5s`. In alternativa possiamo usare i millisecondi, così che mezzo secondo verrà espresso come `500ms`.

Nella Tabella CSS3.2 in Appendice B si elencano le proprietà CSS alle quali è possibile applicare una transizione.

# Animazioni

**TABELLA 15.3** Supporto delle proprietà relative al modulo delle animazioni

REGOLA	BROWSER
@keyframes*	Chrome3+, Safari 3.1+
PROPRIETÀ	BROWSER
animation*	Chrome3+, Safari 3.1+
animation-name*	Chrome3+, Safari 3.1+
animation-duration*	Chrome3+, Safari 3.1+
animation-iteration-count*	Chrome3+, Safari 3.1+
animation-direction*	Chrome3+, Safari 3.1+
animation-delay*	Chrome3+, Safari 3.1+

\* Il supporto è garantito solo per mezzo delle estensioni proprietarie.

Le transizioni rappresentano già di per sé una straordinaria semplificazione di effetti altrimenti accessibili solo per mezzo di funzioni JavaScript sviluppate per conto proprio, o per mezzo di librerie esterne. Tuttavia, se si vuole raggiungere un più alto e raffinato livello di controllo sul tipo di animazione posto in essere tra lo stato iniziale e quello finale, si può ricorrere alle animazioni.

Pur essendo rappresentate da un modulo CSS separato, le animazioni presentano una certa affinità con il modulo CSS delle transizioni, di cui sono un'estensione. La specifica relativa alle animazioni esprime bene la relazione tra le due funzionalità: "Le animazioni sono simili alle transizioni, nel senso che modificano nel tempo il valore di proprietà CSS". La differenza più rilevante risiede nel fatto che mentre le transizioni sono scatenate implicitamente nel momento in cui i valori assegnati alla proprietà cambiano, le animazioni sono esplicitamente eseguite nel momento in cui le proprietà di animazione sono applicate. Questo è il motivo per cui le animazioni richiedono valori esplicativi per le proprietà oggetto di animazione.

Per raggiungere un controllo di più alto livello sulle modalità di svolgimento di una transizione si può ricorrere alla regola `@keyframes`. La sintassi di questa regola è illustrata di seguito:

## **LISTATO 15.7** Regola `@keyframes`: la sintassi

```
@keyframes <identificativo_univoco_animazione> {
```

```
percentuale {  
    /* una o più dichiarazioni css */  
}  
}
```

Una regola `@keyframes` consente di identificare una serie di punti chiave attorno ai quali si svilupperà l'effetto di animazione. Osserviamo come potrebbe essere strutturato un esempio che propone una transizione tra cinque diversi colori:

#### **LISTATO 15.8** Regola `@keyframes`: un esempio

```
@keyframes cinquecolori {  
    0% {  
        color: #3b86ca;  
    }  
    30% {  
        color: #337fab;  
    }  
    60% {  
        color: #3b778c;  
    }  
    90% {  
        color: #67696b;  
    }  
    100% {  
        color: #8e584b;  
    }  
}
```

Questa regola definisce una sorta di linea temporale lungo la quale si snodano cinque diversi set di dichiarazioni. Ogni istante è definito per mezzo di una percentuale (0%, 30%, 60%, 90%, 100%). Dunque questa regola applica il colore `#3b86ca` nel momento iniziale, per poi assumere il colore `#337fab` quando l'animazione sarà al 30% del tempo di esecuzione; via via, questo avvicendamento di colori proseguirà sino al colore `#8e584b`, che sarà proposto come ultimo colore nella sequenza.

#### **NOTA**

In questo momento la regola `@keyframes` è supportata solo da Chrome e Safari e solo con il ricorso al prefisso proprietario, ossia nella denominazione `@-webkit-keyframes`. Qui omettiamo, per comodità, le definizioni di entrambe le regole.

Per il momento iniziale e finale, che deve essere sempre esplicitamente dichiarato, è possibile anche ricorrere alle parole chiave `from` e `to`. In questo modo potremo riscrivere il

listato precedente come segue:

### **LISTATO 15.9** Parole chiave

```
@keyframes cinquecolori {  
  from {  
    color: #3b86ca;  
  }  
  30% {  
    color: #337Fab;  
  }  
  60% {  
    color: #3b778c;  
  }  
  90% {  
    color: #67696b;  
  }  
  to {  
    color: #8e584b;  
  }  
}
```

Attraverso `@keyframes` non solo definiamo una linea temporale lungo la quale una sequenza di proprietà CSS vede attribuirsi valori diversi, ma possiamo stabilire come debba avvenire il passaggio da un punto a quello successivo. Ciò avviene per mezzo della proprietà `animation-timing-function`. Questa proprietà vedrà riconoscere gli stessi valori attribuibili a `transition-timing-function` (Listato 15.4). Il Listato 15.10 ne illustra l'impiego.

### **LISTATO 15.10** Evoluzione dell'animazione con transition-timing-function

```
@keyframes cinquecolori {  
  from {  
    color: #3b86ca;  
    animation-timing-function: ease-out;  
  }  
  30% {  
    color: #337Fab;  
    animation-timing-function: linear;  
  }  
  60% {  
    color: #3b778c;  
    animation-timing-function: linear;  
  }  
  90% {  
    color: #67696b;  
  }  
}
```

```
    animation-timing-function: ease-in;
}
to {
  color: #8e584b;
}
}
```

In questo esempio si impiega un effetto `ease-out` nella transizione che ha luogo tra le dichiarazioni CSS applicate al momento iniziale, contrassegnato dalla parola chiave `from`, e l'istante che intercorre quando il 30% dell'animazione ha avuto luogo. In modo analogo ne consegue che la transizione tra il 30% e il 60% sarà di tipo `linear` ecc.

## animation-name

Una volta definita la regola CSS `@keyframes`, facciamo uso del nome con cui essa è stata identificata univocamente al fine di assegnare tale regola a un elemento.

**LISTATO 15.11** La regola `@keyframes` è associata a un elemento HTML

```
div {
  [omissis]
  -webkit-animation-name: cinquecolori;
  animation-name: cinquecolori;
  [omissis]
}
@keyframes 'cinquecolori' {
  [omissis]
}
```

`animation-name` è la proprietà CSS che realizza il legame tra la regola CSS e l'elemento HTML cui tale regola risulta attribuita. La stessa proprietà `animation-name` può essere utilizzata per associare allo stesso elemento regole diverse.

**LISTATO 15.12** La regola `@keyframes` è associata a un elemento HTML

```
div {
  [omissis]
  -webkit-animation-name: cinquecolori, slittamento;
  animation-name: cinquecolori, slittamento;
  [omissis]
}
@keyframes cinquecolori {
```

```

[omissis]
}

@keyframes slittamento {
  from {
    top: 50px;
    animation-timing-function: ease-out;
  }
  25% {
    top: 150px;
    animation-timing-function: linear;
  }
  50% {
    top: 200px;
    animation-timing-function: linear;
  }
  75% {
    top: 150px;
    animation-timing-function: ease-in;
  }
  to {
    top: 300px;
  }
}

```

In questo caso entrambe le regole di animazione, `cinquecolori` e `slittamento`, risultano applicate allo stesso elemento.

## animation-duration

Per mezzo della proprietà `animation-duration` si determina l'intervallo di tempo, espresso in secondi, lungo il quale si svolge l'animazione.

**LISTATO 15.13** Proprietà `animation-duration`, la durata dell'animazione: un esempio

```

div {
  [omissis]
  -webkit-animation-name: cinquecolori, slittamento;
  animation-name: cinquecolori, slittamento;
  -webkit-animation-duration: 10s;
  animation-duration: 10s;
  [omissis]
}
[omissis]

```

animation-name e `animation-duration` rappresentano il set minimo di proprietà utili a definire un'animazione. Le proprietà che seguono, pur non essendo indispensabili, consentono di aggiungere un ulteriore grado di controllo e personalizzazione sull'effetto di animazione che si intende ottenere.

## animation-iteration-count

Ogni effetto di animazione ha luogo una sola volta, a meno che non si imposti un valore diverso da 1, che è quello predefinito per la proprietà `animation-iteration-count`. Nel listato che segue l'animazione verrà ripetuta per tre volte.

### **LISTATO 15.15** Proprietà `animation-iteration-count`, il numero di iterazioni: un esempio

```
div {  
    [omissis]  
    -webkit-animation-name: cinquecolori, slittamento;  
    animation-name: cinquecolori, slittamento;  
    -webkit-animation-duration: 10s;  
    animation-duration: 10s;  
    -webkit-animation-iteration-count: 3;  
    animation-iteration-count: 3;  
    [omissis]  
}  
[omissis]
```

## animation-direction

Se si esegue l'animazione in più iterazioni si potrebbe optare per una esecuzione alternata: dal principio alla fine e, proseguendo, dalla fine, nuovamente sino al principio.

### **LISTATO 15.16** Proprietà `animation-direction`, il verso dell'animazione: un esempio

```
div {  
    [omissis]  
    -webkit-animation-name: cinquecolori, slittamento;  
    animation-name: cinquecolori, slittamento;  
    -webkit-animation-duration: 10s;  
    animation-duration: 10s;  
    -webkit-animation-iteration-count: 3;  
    animation-iteration-count: 3;  
    -webkit-animation-direction: alternate;  
    animation-direction: alternate;  
    [omissis]
```

}

[omissis]

In questo caso si è optato per un valore diverso da quello predefinito, ossia `normal`. Le iterazioni dispari procedono dal principio alla fine, le iterazioni pari si muovono in senso opposto, dalla fine al principio.

## animation-delay

L'istante in cui ha inizio l'animazione può essere ritardato ricorrendo alla proprietà `animation-delay`. Il valore predefinito di questa proprietà è pari a 0, con il quale l'animazione ha inizio immediatamente, ma è possibile posticiparne l'esecuzione indicando un intero positivo. Se si applica invece un valore negativo, l'animazione ha luogo immediatamente. Tuttavia, essa non inizia dal principio ma in media res, ossia ad animazione già iniziata, per una durata pari al numero di secondi attribuiti a tale proprietà.

### **LISTATO 15.16** Proprietà `animation-delay`, esecuzione ritardata di un'animazione: un esempio

```
div {
    [omissis]
    -webkit-animation-name: cinquecolori, slittamento;
    animation-name: cinquecolori, slittamento;
    -webkit-animation-duration: 10s;
    animation-duration: 10s;
    -webkit-animation-iteration-count: 3;
    animation-iteration-count: 3;
    -webkit-animation-direction: alternate;
    animation-direction: alternate;
    -webkit-animation-delay: 1s;
    animation-delay: 1s;
}
[omissis]
```

### **animation**

Una volta acquisita una certa dimestichezza con queste proprietà, tornerà utile una forma sintattica contratta accessibile per mezzo della proprietà `animation`.

### **LISTATO 15.17** Proprietà `animation`: la sintassi

```
animation: <animation-name> <animation-duration> <animation-timing-function>
```

```
<animation-delay> <animation-iteration-count> <animation-direction>
```

Questa sequenza di proprietà può essere ripetuta tante volte quante sono le animazioni che si vogliono attribuire a un dato elemento. Nel Listato 15.18 utilizziamo la proprietà `animation` per impostare due distinte animazioni:

#### **LISTATO 15.18** Proprietà `animation`, due animazioni: la sintassi

```
-webkit-animation: cinquecolori 10s 1s 3 alternate, slittamento 10s 1s 3 alternate;  
animation: cinquecolori 10s 1s 3 alternate, slittamento 10s 1s 3 alternate;
```

Potremmo dunque riscrivere il Listato 15.16 come segue:

#### **LISTATO 15.19** Proprietà `animation`: un esempio

```
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset="utf-8">  
    <title>Animazioni</title>  
    <style>  
      div {  
        margin: 50px;  
        padding: 1em;  
        width: 400px;  
        border: thin dotted black;  
        font: bold 1.3em/1.3em arial, verdana, sans-serif;  
        position: absolute;  
        -webkit-animation: cinquecolori 10s 1s 3 alternate, slittamento 10s 1s 3 alternate;  
        animation: cinquecolori 10s 1s 3 alternate, slittamento 10s 1s 3 alternate;  
      }  
      @-webkit-keyframes cinquecolori {  
        from {  
          color: #3b86ca;  
          animation-timing-function: ease-out;  
        }  
        30% {  
          color: #337Fab;  
          animation-timing-function: linear;  
        }  
        60% {  
          color: #3b778c;  
          animation-timing-function: linear;  
        }  
      }
```

```

}

90% {
  color: #67696b;
  animation-timing-function: ease-in;
}

to {
  color: #8e584b;
}

}

@-webkit-keyframes slittamento {

from {
  top: 50px;
  animation-timing-function: ease-out;
}

25% {
  top: 150px;
  animation-timing-function: linear;
}

50% {
  top: 200px;
  animation-timing-function: linear;
}

75% {
  top: 150px;
  animation-timing-function: ease-in;
}

to {
  top: 300px;
}

}

</style>
<script src="modernizr-1.6.min.js"></script>
</head>
<body>
<div>
  Lorem ipsum dolor sit amet [omissis].
</div>
</body>
</html>

```

Con questo listato otteniamo un blocco `<div>` che, pur ondeggiando, slitta progressivamente verso il basso mentre il testo cambia colore assumendo cinque diverse sfumature. Sebbene di scarsa utilità pratica, il listato ha lo scopo di illustrare il funzionamento e il campo di applicazione di ciascuna proprietà. Si rimanda ai riferimenti seguenti per prendere visione di alcune applicazioni già esistenti che sfruttano gli effetti

discussi in questo capitolo.

## Riferimenti alle risorse citate e altri link utili

- Modulo CSS di terzo livello relativo alle transizioni: <http://www.w3.org/TR/css3-transitions/>
- Modulo CSS di terzo livello relativo alle animazioni (rappresenta un'estensione del modulo che ha a oggetto le transizioni): <http://www.w3.org/TR/css3-animations/>
- Fiocchi di neve si dissolvono non appena giungono in fondo al monitor, il tutto realizzato con ricorso ad animazioni CSS3. La tecnica è illustrata in questo articolo: <http://natbat.net/2010/Jan/15/css3snow/>
- Applicazione pratica che simula il passaggio da una slide alla successiva con effetto zoom e dissolvenza. L'effetto, realizzato per mezzo delle animazioni CSS3, è illustrato a questo indirizzo: <http://erik.eae.net/archives/2009/02/20/17.47.41/>
- Splendido cartone animato di Spider-Man realizzato con ricorso a trasformazioni e animazioni, in uno stile che ai meno giovani farà tornare alla mente l'omonimo cartone trasmesso da Super Gulp in onda sulla Rai a fine anni Settanta: <http://www.optimum7.com/internet-marketing/web-development/pure-css3-spiderman-ipad-cartoonjquery-html5-no-flash.html>

## Conclusioni

Mai come in questo capitolo i link proposti a corredo come approfondimento costituiscono un necessario complemento al materiale illustrato nel libro. Infatti, animazioni e transizioni non si prestano molto bene a una efficace loro rappresentazione su carta. Inoltre, le risorse proposte offrono una, seppur limitata, idea della gamma di applicazioni possibili, da quelle dichiaratamente ludiche a quelle più utili nel più ordinario lavoro d'ufficio. È facile, a questo punto dell'evoluzione e del supporto di questa tecnologia, puntare il dito contro le sue fragilità. I punti deboli sono ancora evidenti; li possiamo sintetizzare in: supporto limitato e mancanza di strumenti che possano essere d'aiuto nella realizzazione di effetti complessi. Tutto vero. Così come è vero che il supporto non potrà che migliorare di mese in mese con il lento ma progressivo rinnovo a versioni più recenti dei browser più diffusi. In merito al grado di maturazione della tecnologia, bisogna ricordare che tutte le tecnologie web hanno sinora dimostrato una rapida e straordinaria capacità di evoluzione.

Adattarsi al minimo comune denominatore di funzionalità riconosciute da tutti i browser non è una strategia che paga nel lungo periodo, perché spesso si è dimostrata la strada più rapida verso l'obsolescenza. La sfida che pongono le tecnologie legate al Web è quella di offrire funzionalità che possano realmente migliorare l'esperienza d'uso dei visitatori senza pregiudizio per chi non è interessato a investire tempo in un periodico aggiornamento del proprio browser. In quest'ottica, le nuove funzionalità introdotte con i

moduli CSS di terzo livello sono decisamente entusiasmanti.

# Appendice A

## HTML5: tabella di supporto

La seguente tabella illustra il grado di supporto di elementi e attributi HTML5 oggetto di discussione nella prima parte di questo manuale. È bene tenere a mente che si tratta di una lista incompleta di tutti i nuovi elementi e relativi attributi introdotti con questa versione del linguaggio di marcatura.

La dicitura "Supp. parz." indica una implementazione non completa della funzionalità.

**TABELLA HTML5.1** Tabella analitica di supporto di elementi e attributi discussi nel manuale

	CHROME	FIREFOX	INTERNET EXPLORER	OPERA	SAFARI
<b>ELEMENTI</b>					
<article>...</article>	5+	4	9	10.1+	3.2
<aside>...</aside>	5+	4	9	10.1+	3.2
<audio>...</audio>	3+	3.5+	9	10.5+	3.2+
<canvas>...</canvas>	3+	1.5+	9	9+	2+
<datalist>...</datalist>				9+	
<figcaption>...</figcaption>		4			
<figure>...</figure>		4			
<footer>...</footer>	5+	4	9	10.1+	3.2
<header>...</header>	5+	4	9	10.1+	3.2
<hgroup>...</hgroup>	5+	4	9	10.1+	3.2
<keygen>	1	1+		3+	1.2+
<nav>...</nav>	5+	4	9	10.1+	3.2
<mark>...</mark>	7	4	9		
<meter>...</meter>	5+				
<output>...</output>				9+	
<progress>...</progress>	6+				

<section>...</section>	5+	4	9	10.1+	3.2
<rt>...</rt>	5+		5.5+		5+
<time>...</time>	Nessun supporto				
<video>...</video>	3+	3.5+	9	10.5+	3.2+

## FORMS: TIPI DI INPUT

color	Nessun supporto				
date	Supp. parz.			9+	
datetime	Supp. parz.			9+	
datetime-local	Supp. parz.			9+	
email	5+	4		9+	4+
month	Supp. parz.			9+	
number	5+			9+	
range	5+			9+	3.2+
search	5+	4			3.2+
tel	5+	4		10.6+	4+
time	Supp. parz.			9+	
url	5+	4		9+	4+
week	Supp. parz.			9+	

## FORMS: ATTRIBUTI

autofocus	5+	4		10+	4+
autocomplete		4		9+	
list		4		9+	5+
max	5+			9+	5+
min	5+			9+	5+
pattern	5+	4		9+	5+
placeholder	5+	4			4+
required	5+	4		9+	5+

step	5+			9+	5+
------	----	--	--	----	----

## INTERFACCE DI PROGRAMMAZIONE (API)

Applicazioni web offline	4+	3.5+		10.6+	4+
Canvas 2D API	4+	3.5+		10.6+	4+
Geolocalizzazione API	5+	3.5+		10.6+	5+
Multimedia API	4+	3.5+	9	10+	4+
Web Storage	4+	3+	8+	10.5+	4+

# Appendice B

## CSS3: tabella di supporto

La Tabella CSS3.1 illustra il supporto ai selettori, proprietà e regole discussi nella seconda parte del capitolo. La Tabella CSS3.2 propone un elenco di tutte le proprietà cui possa essere applicato un effetto di transizione.

**TABELLA CSS3.1** Supporto delle principali proprietà e selettori trattati nel manuale

	<b>CHROME</b>	<b>FIREFOX</b>	<b>INTERNET EXPLORER</b>	<b>OPERA</b>	<b>SAFARI</b>
<b>SELETTORI</b>					
selettori ~	4+	3+	7+	10.10+	4+
[attr]selettori	4+	3+	7+	10.10+	4+
:empty	4+	3+	9	10.10+	4+
:enabled :disabled :checked	4+	3+	9	10.10+	4+
:first-of-type	4+	3.5+	9	10.10+	4+
:last-child	4+	3.5+	9	10.10+	4+
:last-of-type	4+	3.5+	9	10.10+	4+
:not	4+	3+	9	10.10+	4+
:nth-child()	4+	3.5+	9	-	4+
:nth-last-child()	4+	3.5+	9	10.10	4+
:nth-last-of-type()	4+	3.5+	9	10.10	4+
:nth-of-type()	4+	3.5+	9	-	4+
:only-child	4+	3+	9	10.10	4+
:only-of-type	4+	3.5+	9	10.10	4+
:root	4+	3+	9	10.10	4+
::selection	4+	1.5 (-moz-)	9	10.10	4+
:target	4+	3+	9	-	4+

# PROPRIETÀ

animation-name	3+ (-webkit-)				3.1+ (-webkit-)
animation-duration	3+ (-webkit-)				3.1+ (-webkit-)
animation-iteration-count	3+ (-webkit-)				3.1+ (-webkit-)
animation-direction	3+ (-webkit-)				3.1+ (-webkit-)
animation-delay	3+ (-webkit-)				3.1+ (-webkit-)
background-clip	3+ (-webkit-)	3.5+ (-moz-)	Supp. err.	10.5+	5 (-webkit-)
background-origin	3+ (-webkit-)	3.5+ (-moz-)	Supp. err.	10.5+	5 (-webkit-)
background-size	3+ (-webkit-)	3.6+ (-moz-)	9	9.5+ (-o-)	5 (-webkit-)
border-image	2+ (-webkit-)	3.5+ (-moz-)		10.5+	5 (-webkit-)
border-radius	4+	3.5+ (-moz-)	9	10.5+	5+
border-bottom-left-radius	4+	3.5+ (-moz-)	9	10.5+	5+
border-bottom-right-radius	4+	3.5+ (-moz-)	9	10.5+	5+
border-top-left-radius	4+	3.5+ (-moz-)	9	10.5+	5+
border-top-right-radius	4+	3.5+ (-moz-)	9	10.5+	5+
box-shadow	4+ (-webkit-)	4	9	10.5+	3+ (-webkit-)
columns	4+ (-webkit-)	3+ (-moz-)			4+ (-webkit-)
column-gap	4+ (-webkit-)	3+ (-moz-)			4+ (-webkit-)
column-rule	4+ (-webkit-)	3+ (-moz-)			4+ (-webkit-)
column-width	4+ (-webkit-)	3+ (-moz-)			4+ (-webkit-)
opacity	1+	0.9+	4+ (filter pr.)	9+	1.2
text-overflow	2+		6+	10.7	1.3+
text-shadow	2+	3.5+		9.5+	4+
transform	2+ (-webkit-)	3.5+ (-moz-)	9	10.5+ (-o-)	3.1+ (-webkit-)
transition-property	3+ (-webkit-)	4 (-moz-)		10.5+ (-o-)	3.1+ (-webkit-)
transition-duration	3+ (-webkit-)	4 (-moz-)		10.5+ (-o-)	3.1+ (-webkit-)
transition-timing-function	3+ (-webkit-)	4 (-moz-)		10.5+ (-o-)	3.1+ (-webkit-)

transition-delay	3+ (-webkit-)	4 (-moz-)		10.5+ (-o-)	3.1+ (-webkit-)
transform-origin	2+ (-webkit-)	3.5+ (-moz-)	9	10.5+ (-o-)	3.1+ (-webkit-)
word-wrap	Sì	3.5+	5.5+	10.5+	Sì

## REGOLE

@font-face	5+	3.5+	4+	10.1+	3.1+
@keyframes	3+				3.1+

**TABELLA CSS3.2** Proprietà i cui valori possono essere oggetto di transizione

PROPRIETÀ	VALORE OGGETTO DI TRANSIZIONE
background-color	Colore
background-image	Solo gradienti
background-position	Percentuale, lunghezza (in valore assoluto)
border-bottom-color	Colore
border-bottom-width	Lunghezza (in valore assoluto)
border-color	Colore
border-left-color	Colore
border-left-width	Lunghezza (in valore assoluto)
border-right-color	Colore
border-right-width	Lunghezza (in valore assoluto)
border-spacing	Lunghezza (in valore assoluto)
border-top-color	Colore
border-top-width	Lunghezza (in valore assoluto)
border-width	Lunghezza (in valore assoluto)
bottom	Lunghezza (in valore assoluto), percentuale
color	Colore
crop	Definizione di area rettangolare rect (top, right, bottom, left)

font-size	Lunghezza (in valore assoluto), percentuale
font-weight	Numero
grid-*	Vari
height	Lunghezza (in valore assoluto), percentuale
left	Lunghezza (in valore assoluto), percentuale
letter-spacing	Lunghezza (in valore assoluto)
line-height	Numero, lunghezza (in valore assoluto), percentuale
margin-bottom	Lunghezza (in valore assoluto)
margin-left	Lunghezza (in valore assoluto)
<b>PROPRIETÀ</b>	<b>VALORE OGGETTO DI TRANSIZIONE</b>
margin-right	Lunghezza (in valore assoluto)
margin-top	Lunghezza (in valore assoluto)
max-width	Lunghezza (in valore assoluto), percentuale
min-height	Lunghezza (in valore assoluto), percentuale
min-width	Lunghezza (in valore assoluto), percentuale
opacity	Numero
outline-color	Colore
outline-offset	Intero
outline-width	Lunghezza (in valore assoluto)
padding-bottom	Lunghezza (in valore assoluto)
padding-left	Lunghezza (in valore assoluto)
padding-right	Lunghezza (in valore assoluto)
padding-top	Lunghezza (in valore assoluto)
right	Lunghezza (in valore assoluto), percentuale
text-indent	Lunghezza (in valore assoluto), percentuale
text-shadow	Definizione di un'ombra
top	Lunghezza (in valore assoluto), percentuale
vertical-align	Parole-chiave, lunghezza (in valore assoluto), percentuale

visibility	Visibilità
width	Lunghezza (in valore assoluto), percentuale
word-spacing	Lunghezza (in valore assoluto), percentuale
z-index	Intero
zoom	Numero