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

APOGEO

HTML5 E CSS3

Gabriele Gigliotti

APOGEO

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

~

Seguici su Twitter @apogeonline

A Erica e Giordano

Prefazione

HTMLe 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.

HTMLe 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 HTML4.01 e CSS 2.

Fino a oggi.

Perché HTMLe CSS stanno di nuovo evolvendo e qua e là nel Web spuntano i segni di quanto HTML5 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.

HTML5 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 perseguite a fatica attraverso codici complessi e hack.

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

Fabio Brivio
Editor
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 HTML4.01 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 HTMLè 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 HTMLvengono 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 diversiservizi promossi in Rete. Ne esistono sia gratuiti, come browsershots.org, sia a pagamento, come browsercam.com e 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 hobbistico, dove le risorse economiche tendono a zero; in casi come questi, il suggerimento è disollecitare 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 distringa User Agent

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

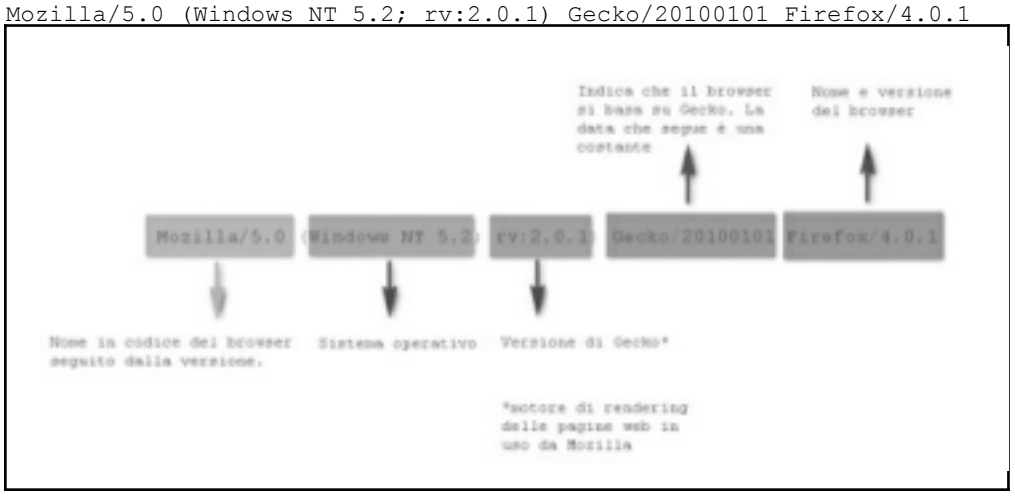

FIGURA 1.1 La stringa User Agent: un condensato di informazioni.

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.

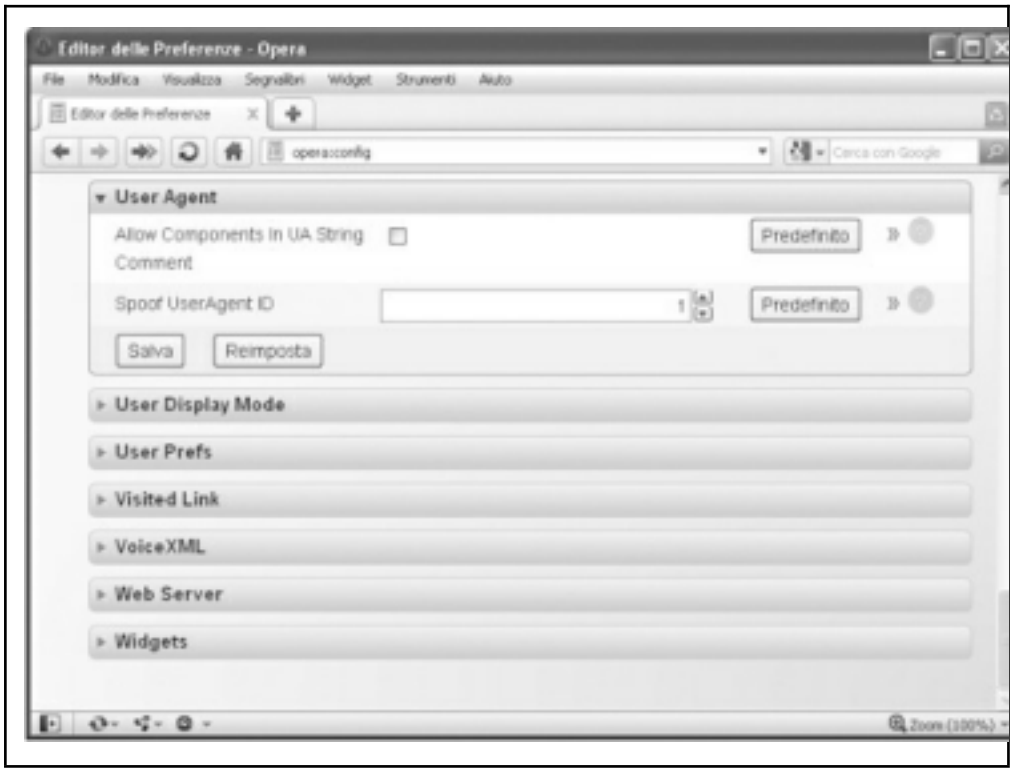

FIGURA 1.2 Scegliendo un valore da 2 a 5 nell'editor delle preferenze di Opera si ottengono altrettante stringhe User Agent.

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 distatistiche 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 necessario tenere conto.

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 disupporto 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 delsupporto 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 delsupporto 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.

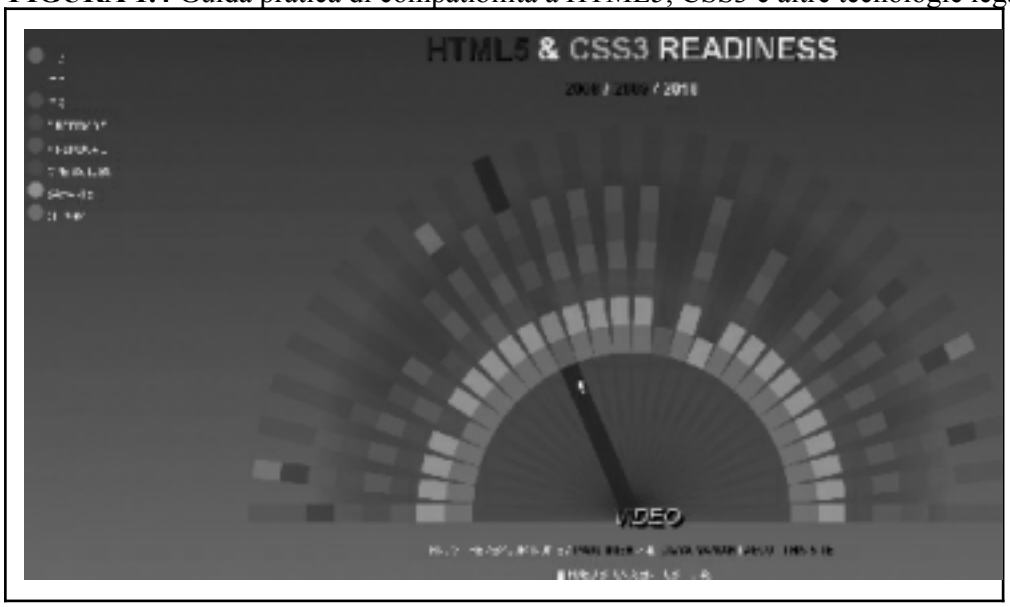

FIGURA 1.5 Un'accattivante rappresentazione grafica del livello disupporto rappresentato in base ai browser più diffusi e all'anno di riferimento.

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

Sicurezza e trattamento dei dati personali

La progressiva diffusione del linguaggio HMTL5 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 isuperpoteri, ha voluto dimostrare come sia possibile creare un sistema di persistenza dei dati realmente efficace. I cookie generati in questo modo saranno salvatisulla 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 versionistabili 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 dei 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 distruttura

ELEMENTO SUPPORTATO	BROWSER
Elementi distruttura	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 HTMLè 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 marcare 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 ipertesti 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 alsolo 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 HTMLmeno 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 ipertesti 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 HTMLidentifica 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 piъ importante di libertp ш di essere ciђ che si ш 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, otterremo qualcosa del tipo: "`Il genere pi&ugrave; importante di libert&agrave; &egrave; di essere ci&ograve; che si &egrave; 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 sisostituirà 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>
  </nav>
  <p>Copyright &copy; 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 distile

```
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 distruttura 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 distruttura in memoria via JavaScript abilita le regole distile 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 distruttura 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>
<img src="museo.jpg" alt="Museo della Scienza di Valenzia">
<figcaption>Veduta esterna del Museo della Scienza di Valenzia 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 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 manua sono in debito e, per dimostrarvi la mia gratitudine, non vi rovinerò la giornata discettando delle sottili differ tra testo da fornire in alternativa alla foto e testo da presentare nella didascalia della foto (se voleste infligge soli queste pene, potreste sempre trovare nella sezione 4.8.1.1.10 “A key part of the content” pane per i vost denti). In casi del genere suggerisco di rifarsi al proprio buon senso rispondendo a una semplice domanda: u utente che ha scelto di disabilitare la visualizzazione delle immagini o un non vedente che usa uno screen re ottiene una descrizione utile alla comprensione dell’immagine? Se la risposta è si, avete assolto al compito d rendere il dato accessibile anche se dal punto di vista puramente semantico potreste aver scelto il posto men ispirato in cui inserire ilsupporto 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 29

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>
```

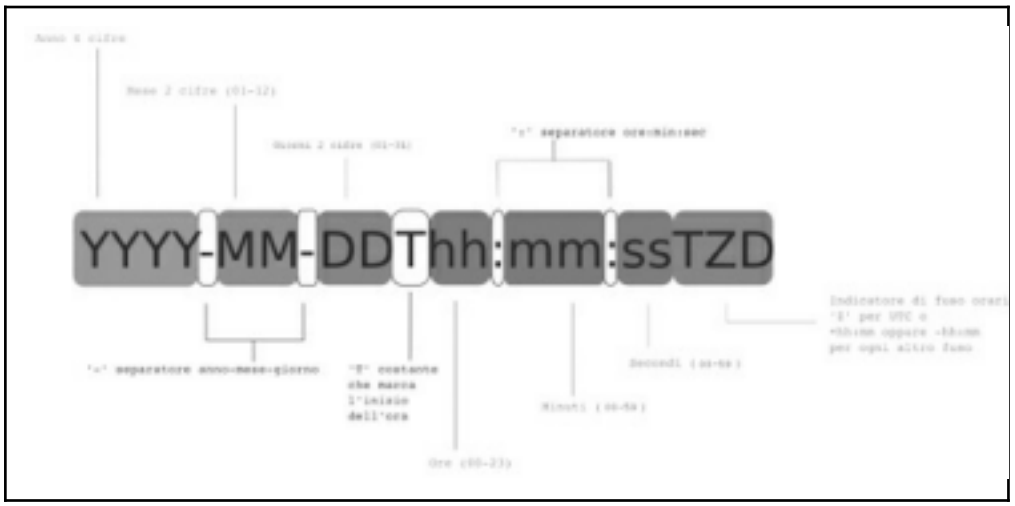

FIGURA 2.2 Elementi del formato ISO data/ora.

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:

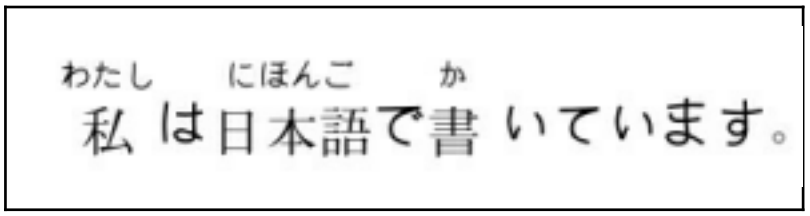

FIGURA 2.3 Kanji e furigana.

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

Per rendere più evidente le sue componenti possiamo rifarci 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 31

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

Su sfondo nero sono rappresentati gli ideogrammi, mentre i furigana a essi relativi sono stati evidenziati in grigio. Per applicare gli elementi HTMLsi 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.

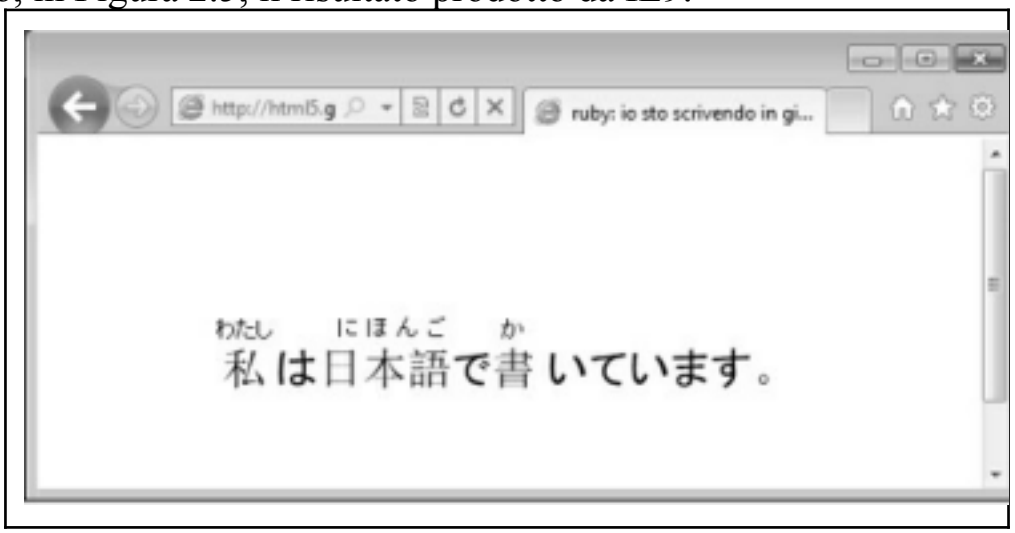

FIGURA 2.5 IE9 beta supporta gli elementi <ruby> e <rt>.

Sfortunatamente, se il browser non supporta questi elementi le annotazioni

ruby non sono 32

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

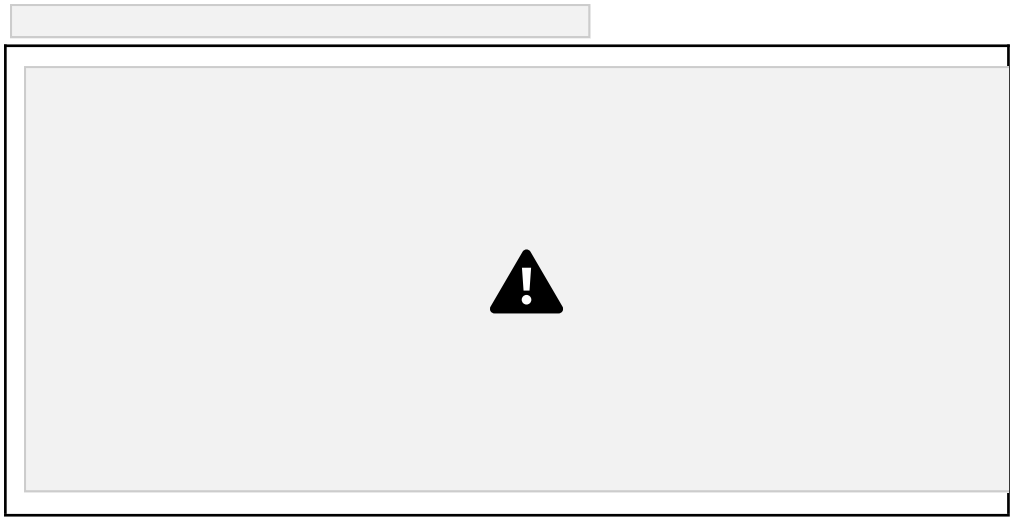

FIGURA 2.6 Il mancato riconoscimento degli elementi porta a un appiattimento dei caratteri, la rappresentazione ne risulta compromessa.

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

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).

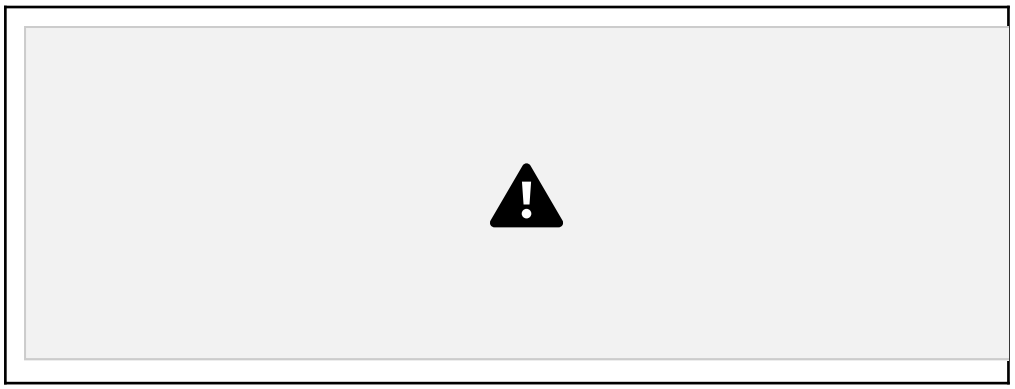

FIGURA 2.7 Le parentesi aiutano a distinguere gli ideogrammi dalle annotazioni ruby.

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).

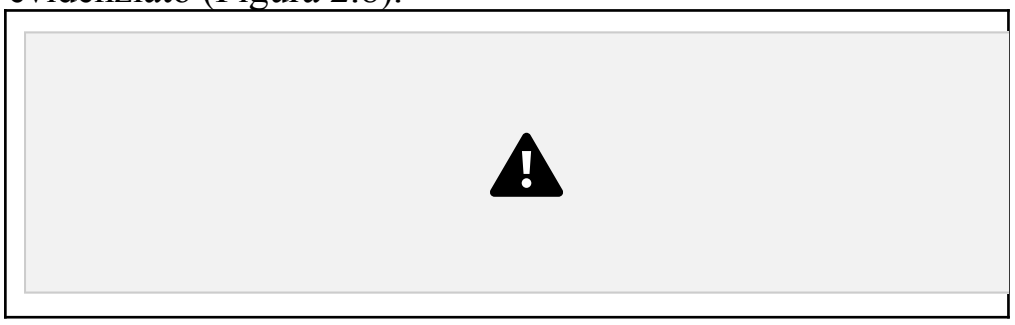

FIGURA 2.8 Chrome visualizza su sfondo giallo il testo contrassegnato con <mark>.

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.

LISTATO 2.23 Enfatizzare uno o più termini con <mark>

```
<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 URLche 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"><img
src="http://images2.corriereobjects.it/Media/Foto/2010/09/11/presi_big.jpg?v=20100911191708"
alt="Tragedia sul lavoro, tre operai morti Napolitano: «Indignato» Foto|Video " title="Tragedia sul
lavoro, tre operai morti Napolitano: «Indignato» Foto|Video"/>
</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>
      <img
src="http://images2.corriereobjects.it/Media/Foto/2010/09/11/presi_big.jpg?v=20100911191708"
alt="Tragedia sul lavoro, tre operai morti Napolitano: «Indignato» Foto|Video " title="Tragedia sul
lavoro, tre operai morti Napolitano: «Indignato» Foto|Video"/>
    </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>`.

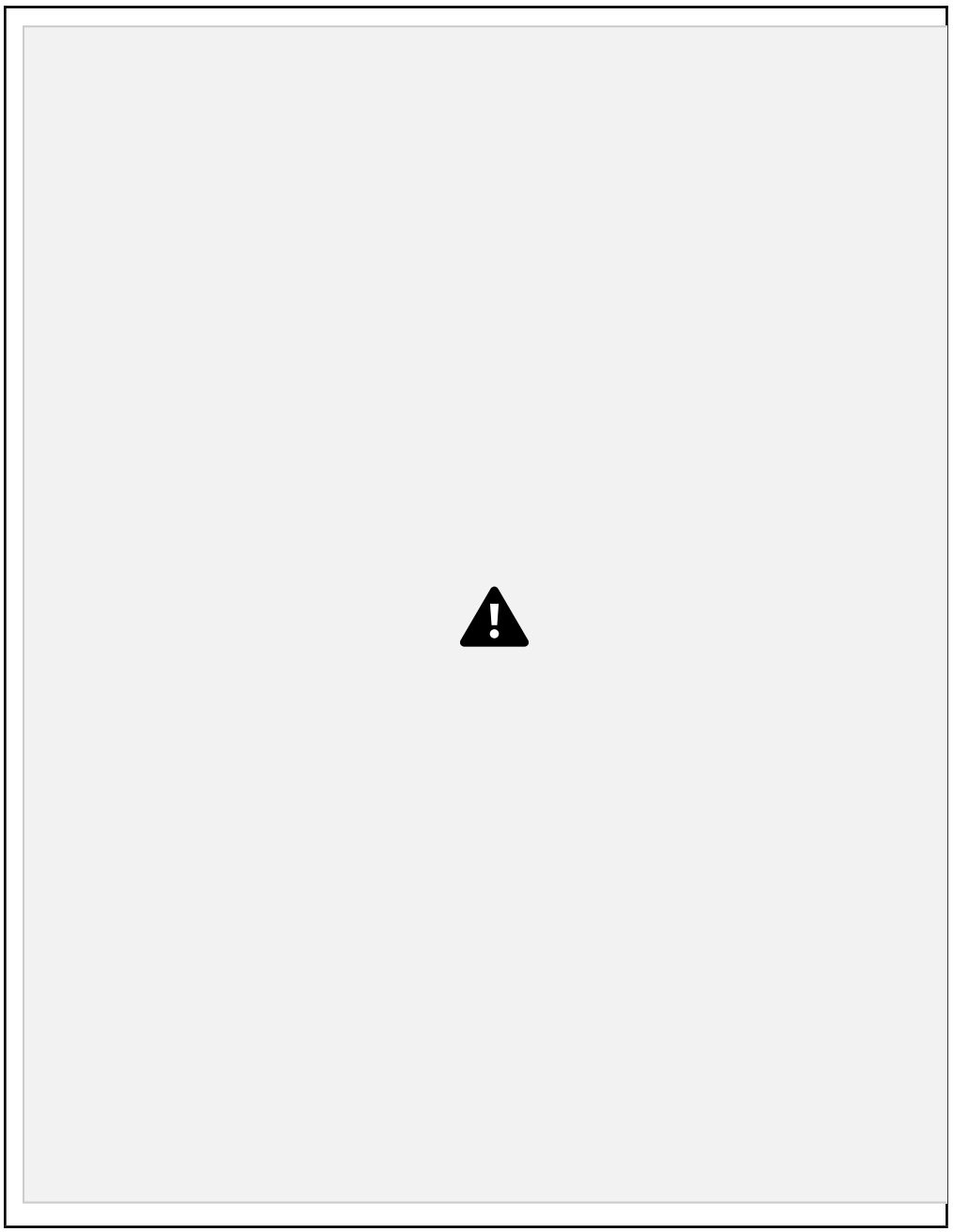

FIGURA 2.9 Sia il titolo sia l'immagine sono contrassegnati da due elementi <a>: entrambi puntano allo stesso documento.

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>&amp;quot;Dove diavolo si trova Via Solferino, 28!&amp;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: trafelato 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)

Ilsolo vero validatore semantico esistente oggi è una persona. Non esiste alcun software in grado distabilire se l'elemento `<address>` sia stato utilizzato in modo più o meno appropriato. Detto questo, dopo aver condotto una rapida analisisulla 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. Cisono altri aspetti certamente più criticisu 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 sforbiciate 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'inchiostro, 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 HTML4.01 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

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

applet	dir	marquee	s
	41		

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 controintuitiva tra funzionalità *obsolete non conformi* e *obsolete ma conformi*). A questo proposito ho trovato divertente il modo spiccio 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 solleticato 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 siate 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.

Capitolo 3

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 URLsi 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"

Maisentito parlare di "Dialogo nel Buio"? È una mostra ospitata nella sede dell'Istituto dei ciechi a Milano. 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 grad 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 46

da quanto accade con altri controlli.

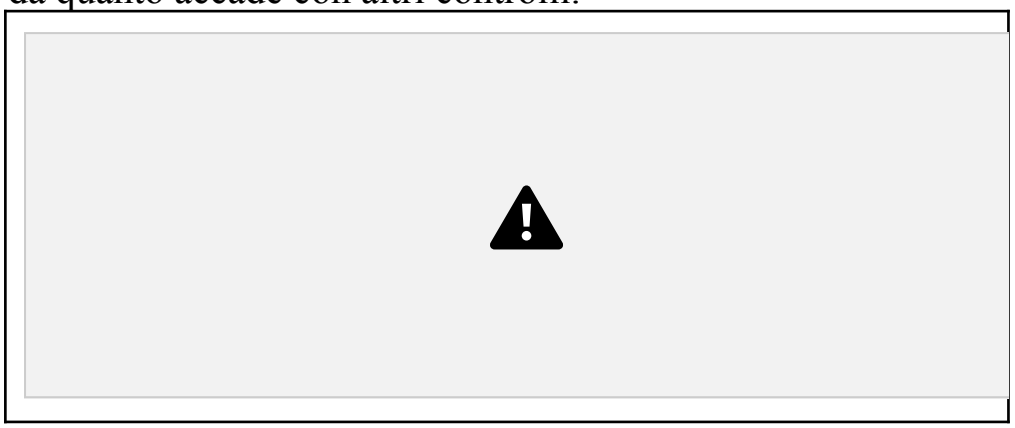

FIGURA 3.1 Campo di ricerca con angoli arrotondati, senza focus.

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

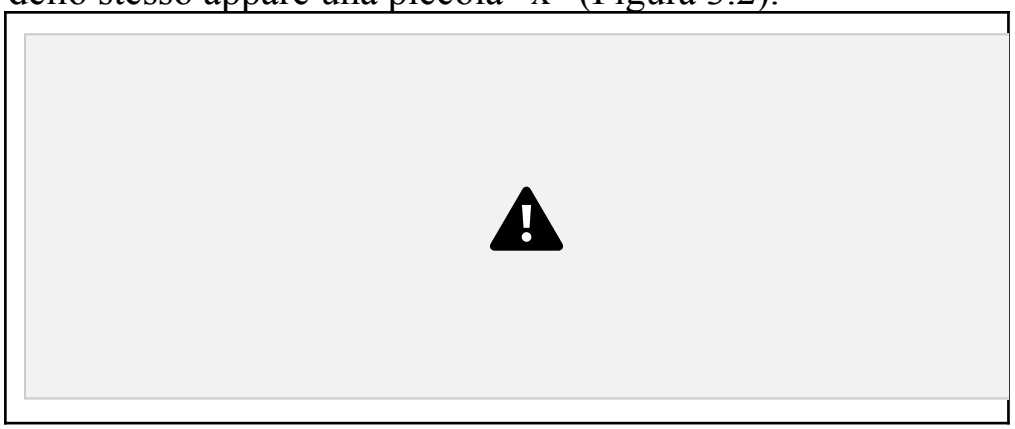

FIGURA 3.2 Nella parte sinistra del campo una piccola "x" funge da pulsante di reset.

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).

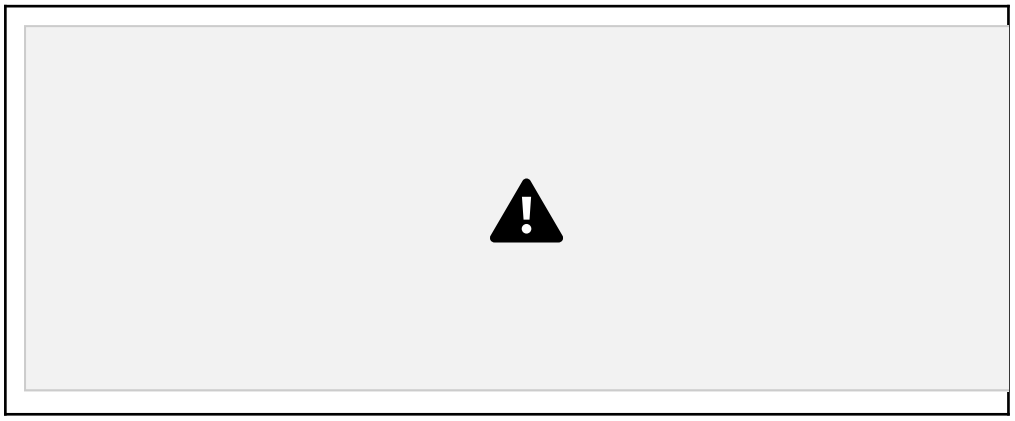

FIGURA 3.3 L'icona di una "x" segnala la possibilità di reimpostare il testo di ricerca.

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>
    <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).

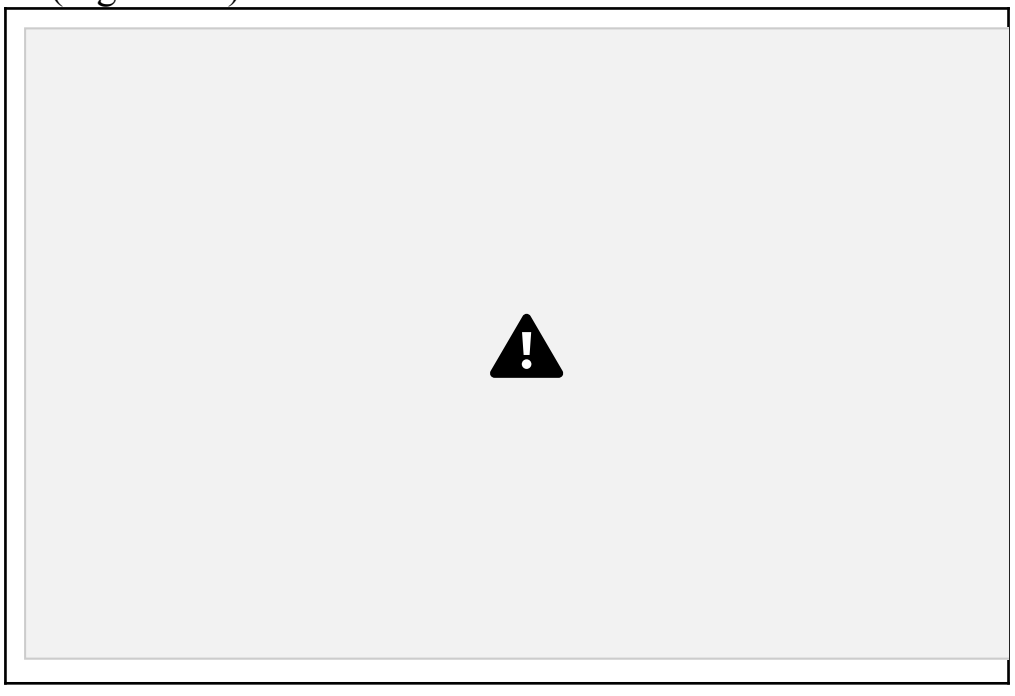

FIGURA 3.4 Opera caratterizza il campo Email con l’icona di una busta da lettera (ma non su Linux!). 49

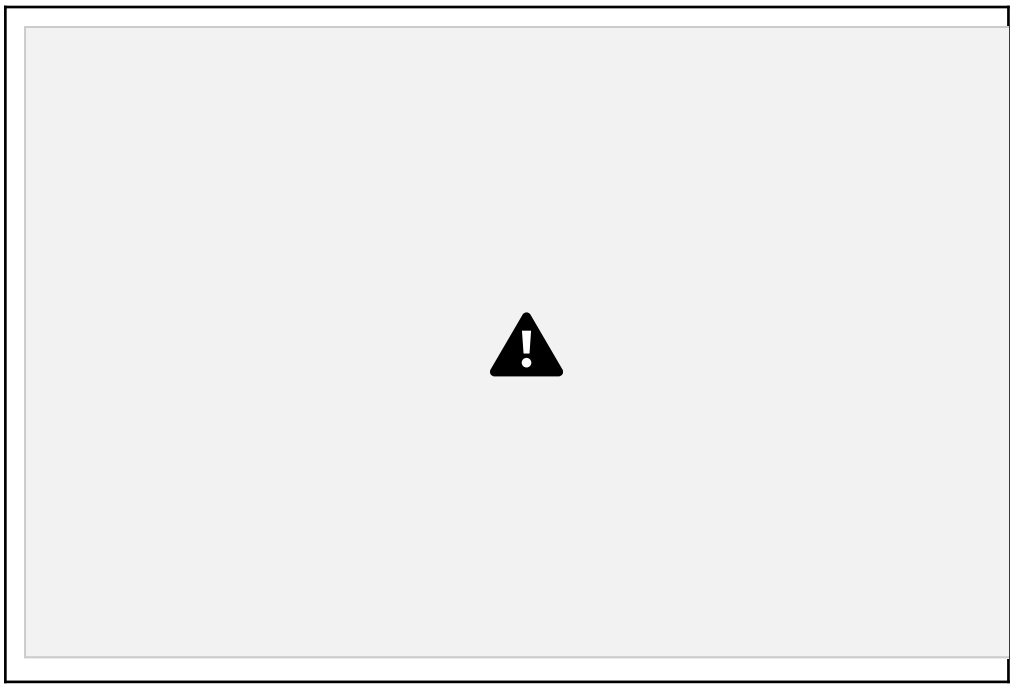

FIGURA 3.5 Opera avvisa l'utente che l'indirizzo inserito non risulta essere valido.

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.

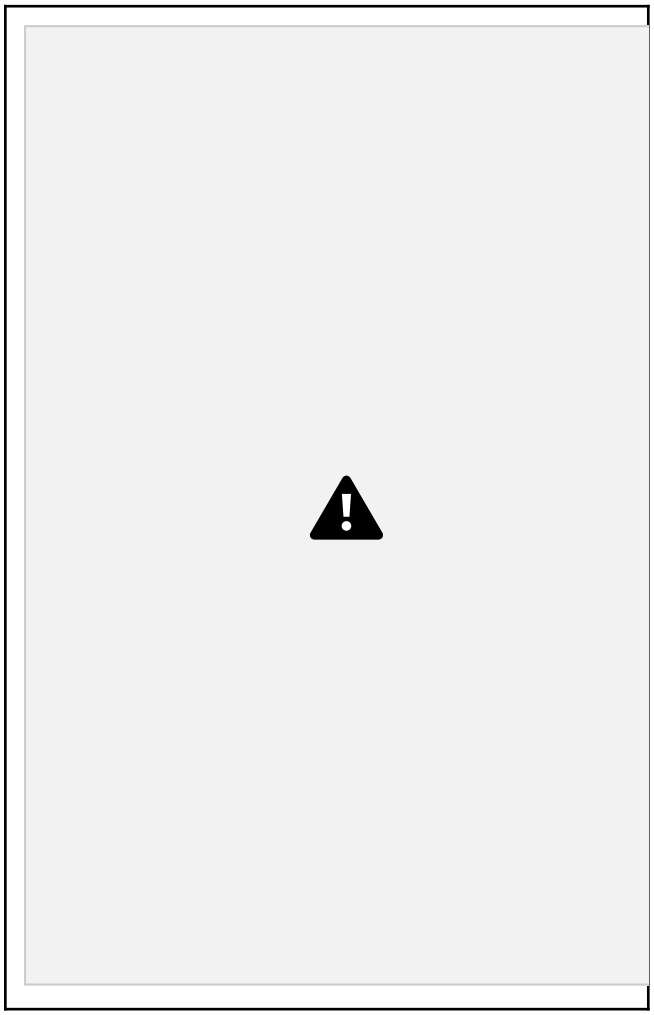

FIGURA 3.6 La tastiera touch proposta da Safari Mobile su iPhone dedicata al campo email.

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 HTMLesistono 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).

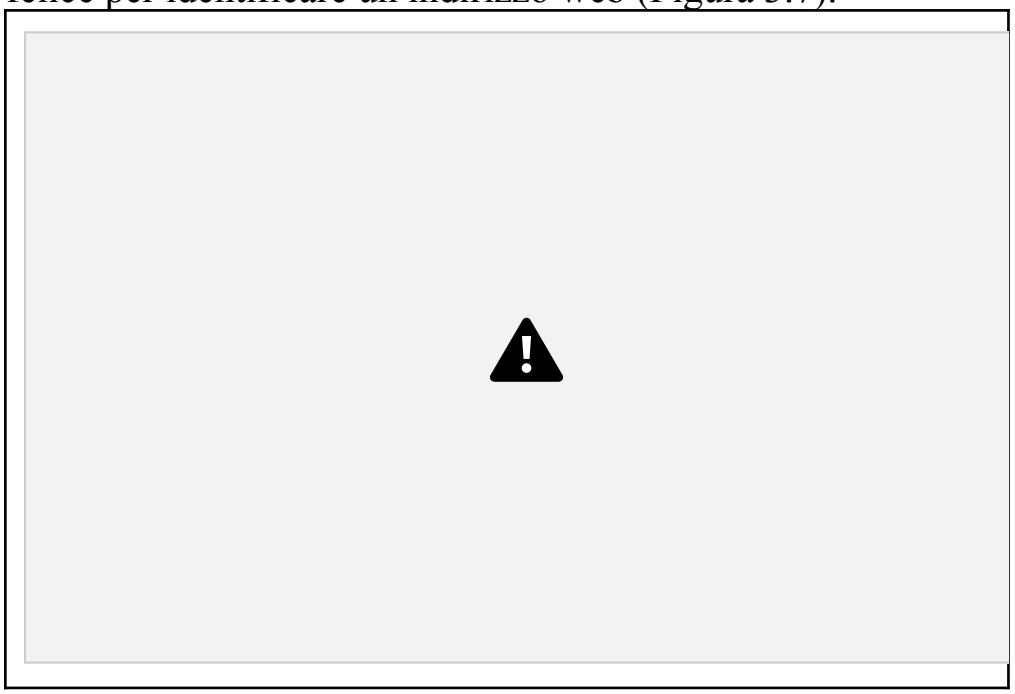

FIGURA 3.7 Caratterizzazione stilistica del campo <input type=”url”> proposta da Opera.

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).

tel

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>
```

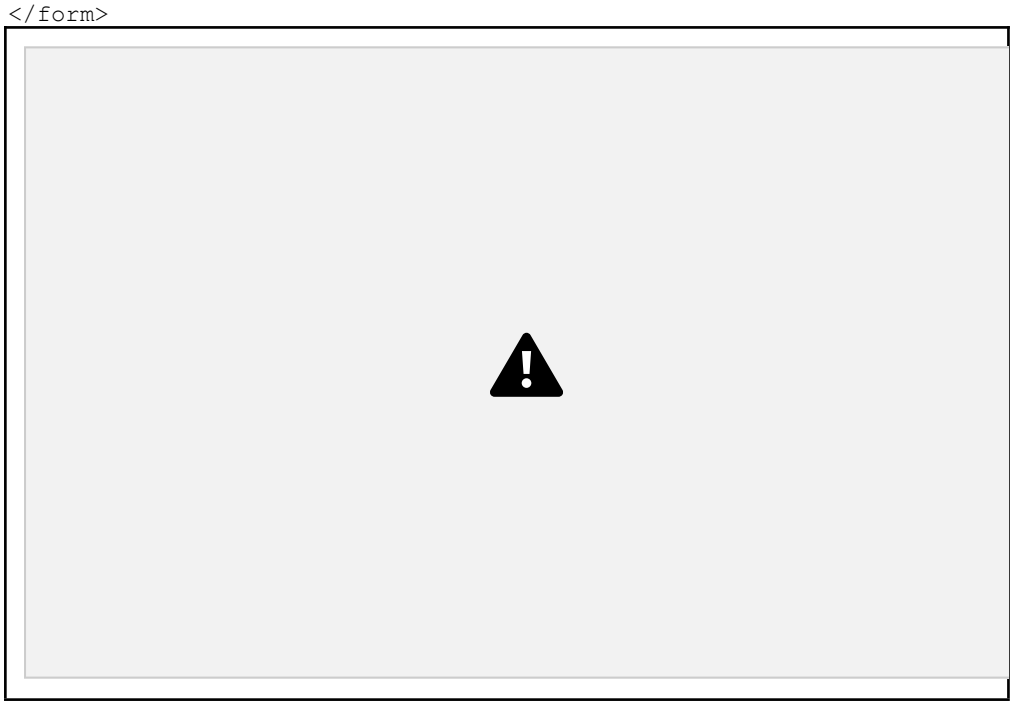

FIGURA 3.8 Opera segnala che l’indirizzo immesso non è valido.

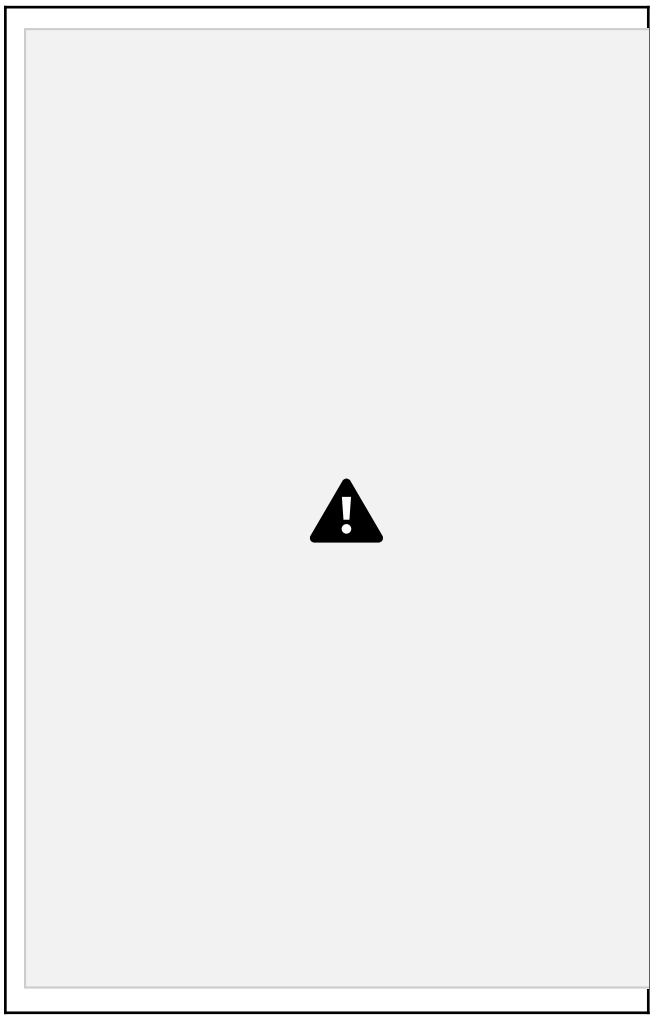

FIGURA 3.9 Tastiera touch dedicata per il campo <input type=”url”>.

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).

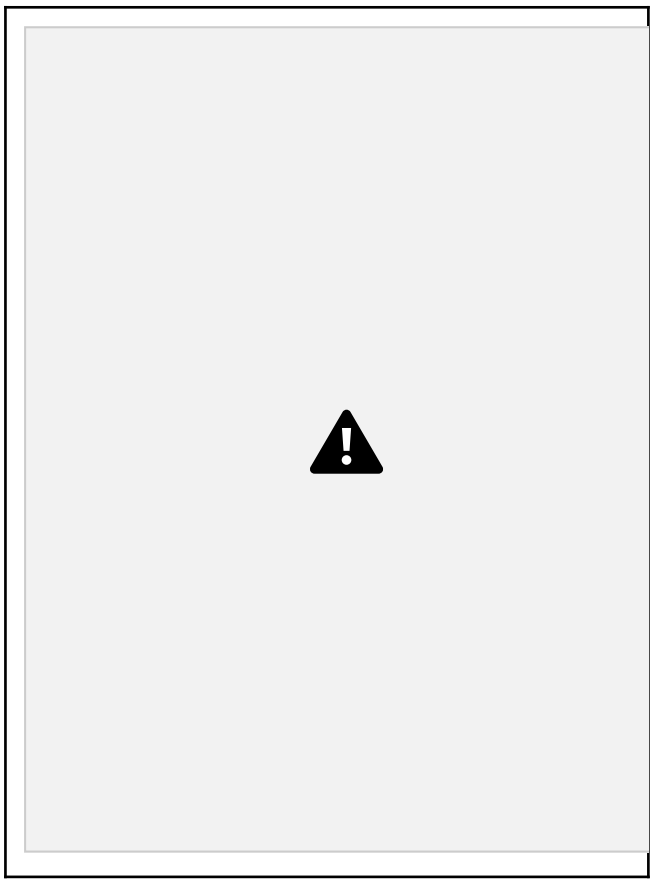

FIGURA 3.10 Tastiera touch dedicata per il campo <input type=”tel”>.

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).

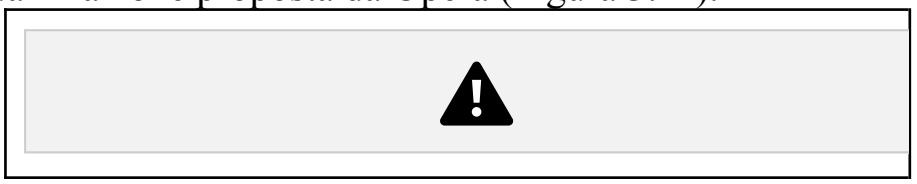

FIGURA 3.11 Lo spinbox: a sinistra nella riproduzione proposta da Opera, a destra secondo il layout di Chrome. 56

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,

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.

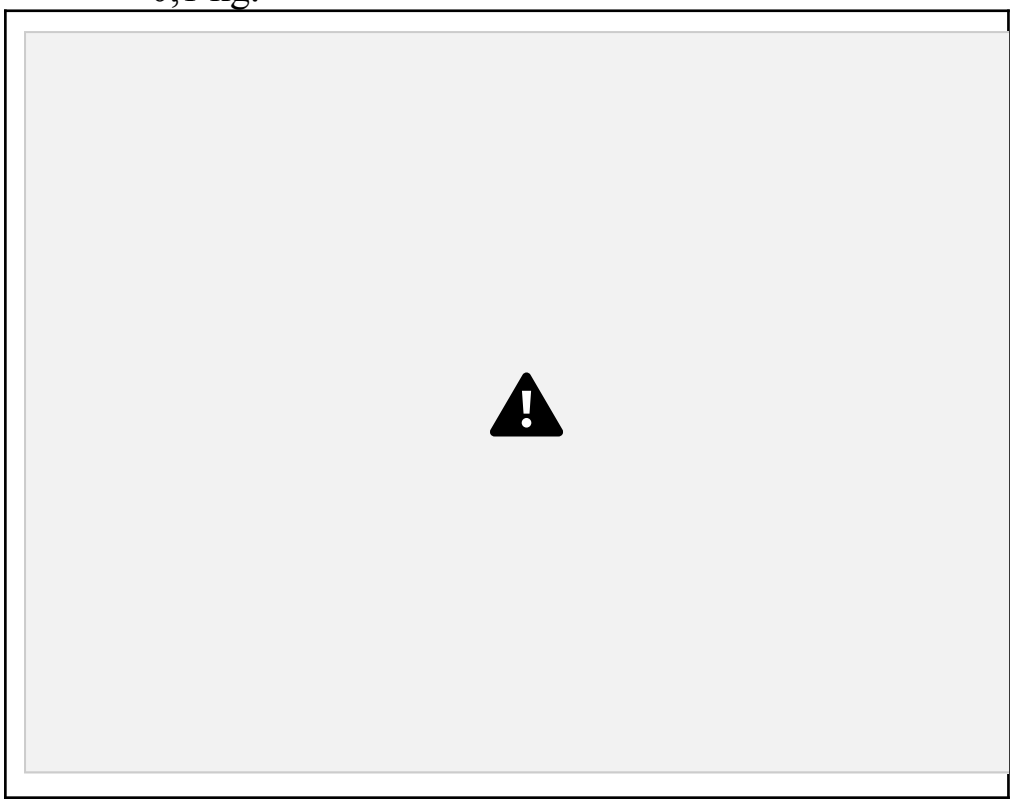

FIGURA 3.12 Campispinbox per l'inserimento di valori numerici.

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.

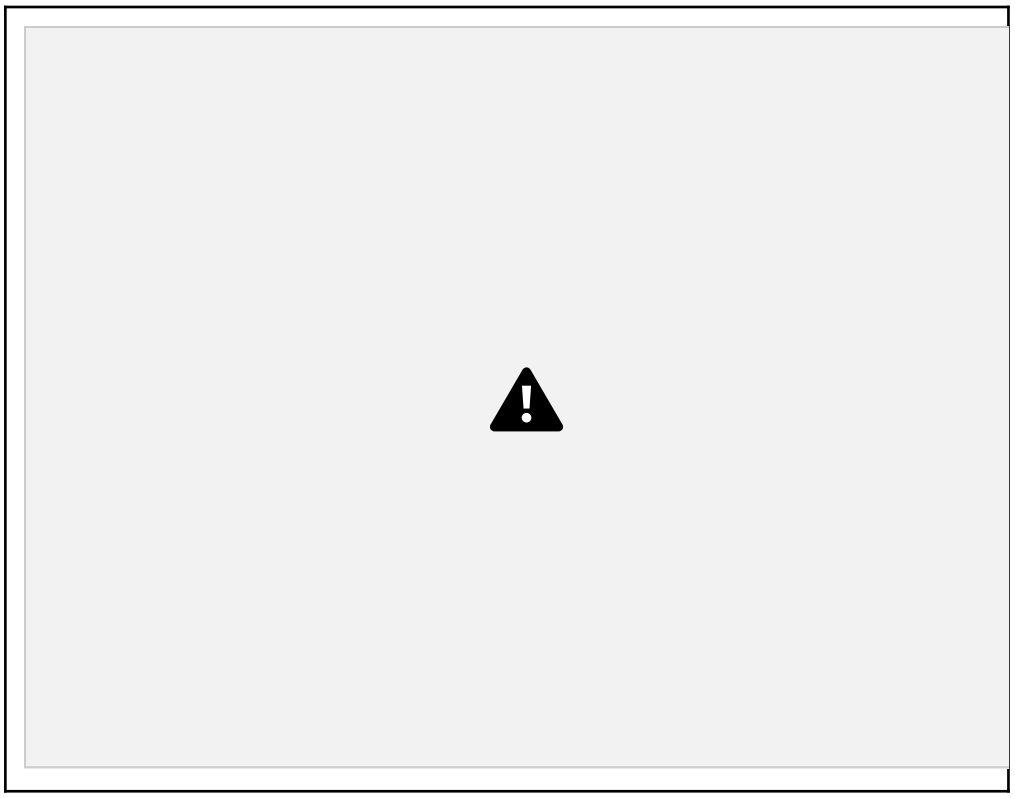

FIGURA 3.13 Opera inibisce l'invio del form se il valore che si propone non soddisfa i vincoli posti dal controllo.

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).

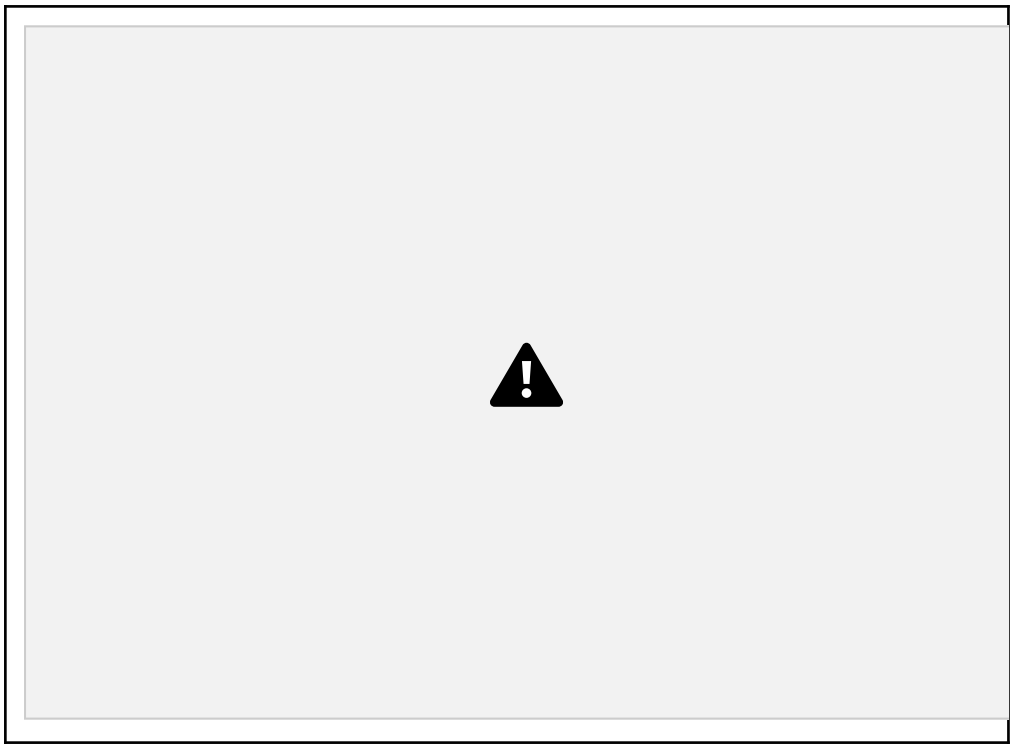

FIGURA 3.14 Chrome soffre di un errore infinitesimale nel calcolo dei valori decimali.

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.

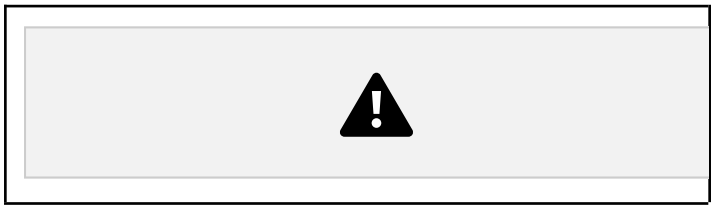

FIGURA 3.16 Lo slider proposto da Safari 5 su XP SP2.

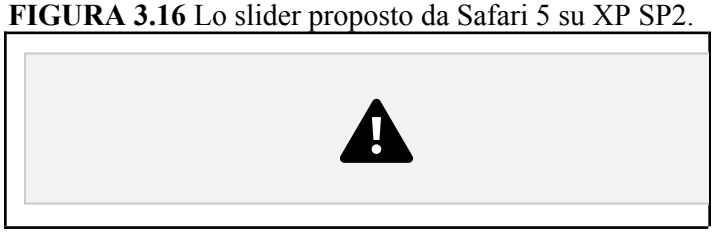

FIGURA 3.17 Lo slider proposto da Safari 5 su Windows 7.

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 disupporto 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.

Capitolo 4

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.)

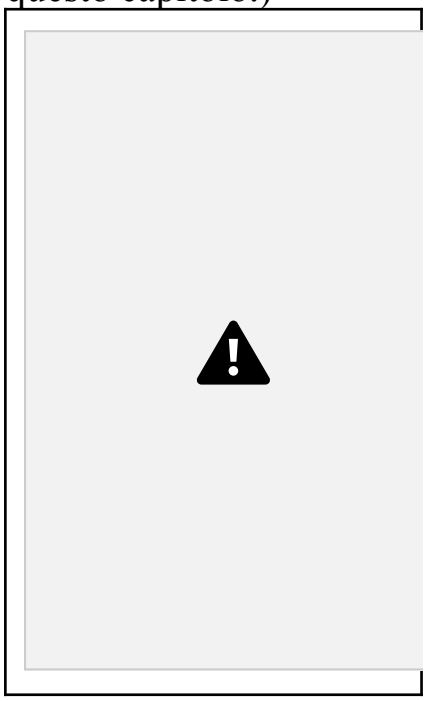

FIGURA 4.1 Se il "senso di ragno" non ci aiuta, passiamo in rassegna la lista deisuggerimenti in coda al capitolo [Uomo Ragno, "Ciudad de las Artes y las Ciencias", Valencia, Spagna]

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.

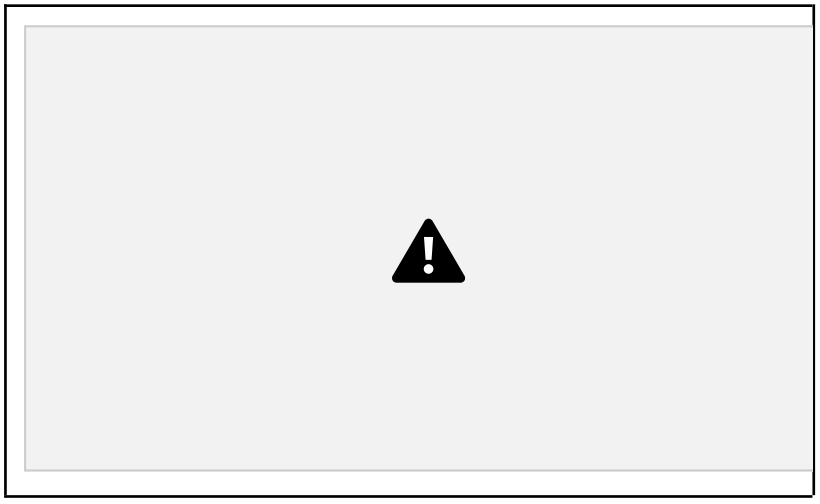

FIGURA 4.2 Il date picker standard com'è proposto da Opera.

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

Se l'attributo `step` non è stato impostato, il suo valore predefinito è pari a 1

(giorno). Si 65

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">
```

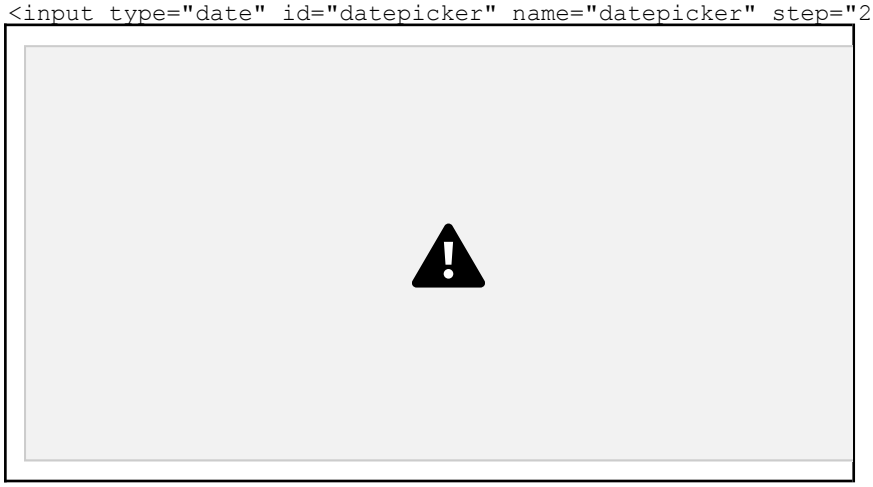

FIGURA 4.3 È possibile limitare in modo altamente personalizzabile il numero di opzioni oggetto discelta.

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 isecondi quando sisceglie 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">
```


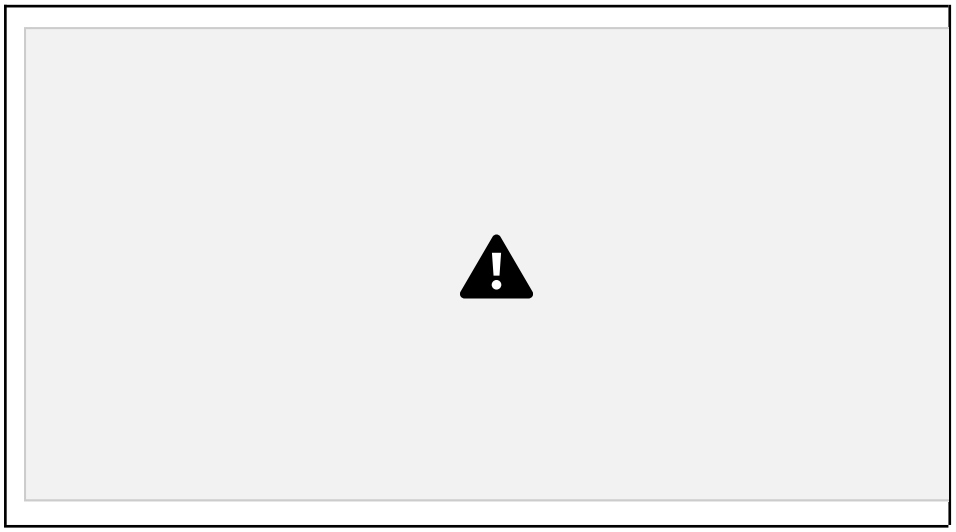

FIGURA 4.6 Impostazione di data e ora con informazionisul fuso orario (UTC).

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.

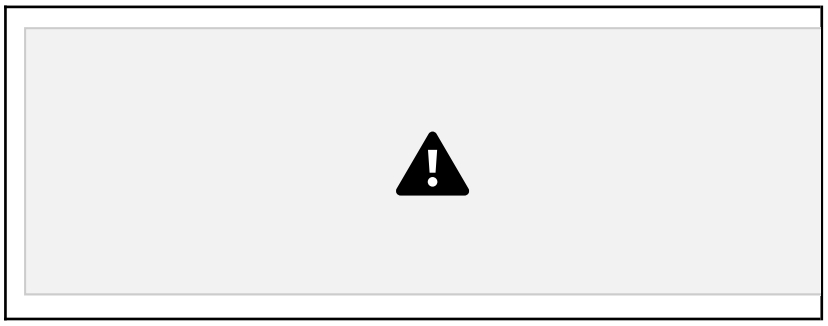

FIGURA 4.7 Nel campo di testo sono visualizzatisolo anno e mese.

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).

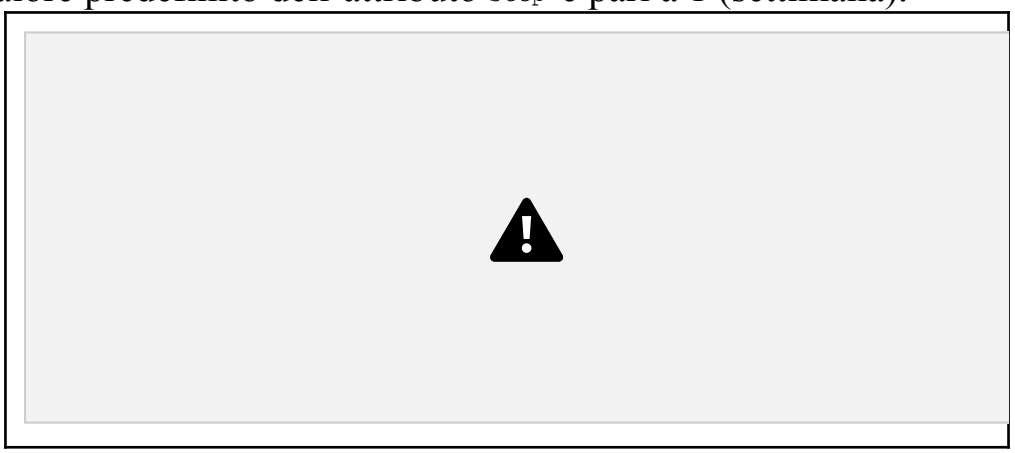

FIGURA 4.8 La selezione della settimana è enfatizzata nel controllo da una striscia a sfondo grigio.

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 69

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.

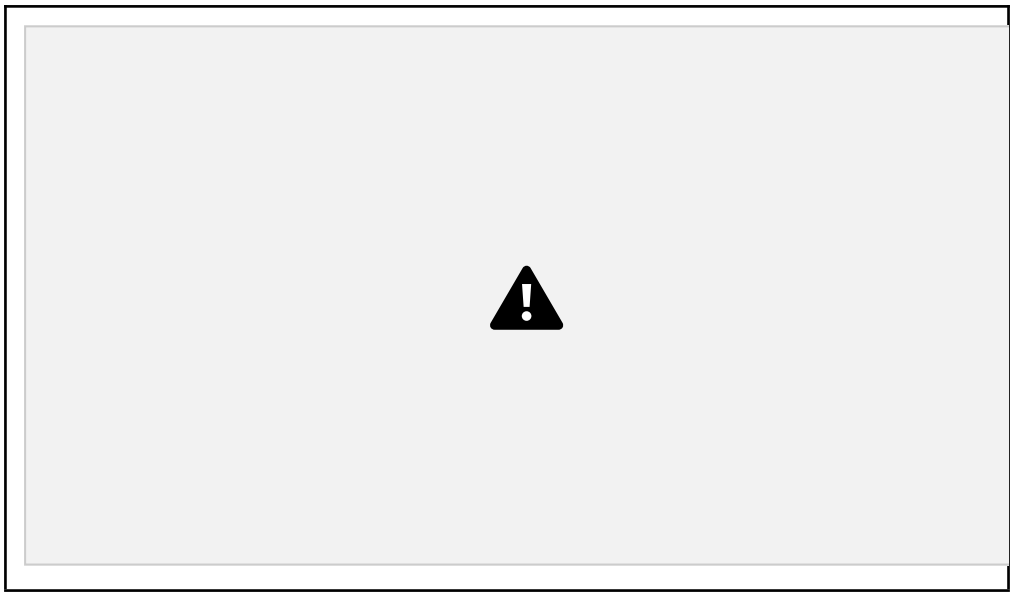

FIGURA 4.9 Il color picker.

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).

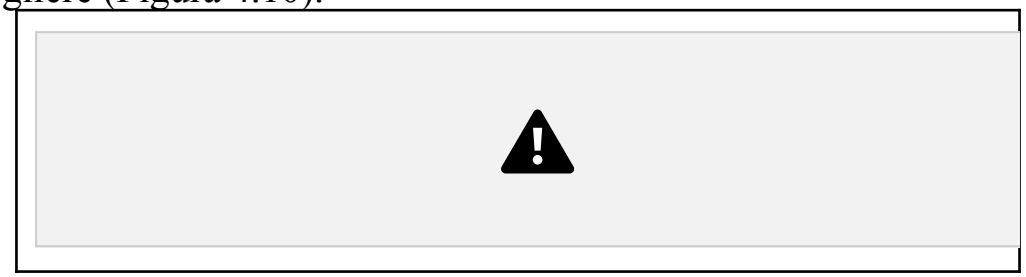

FIGURA 4.10 Il datalist arricchisce il campo di testo offrendo opzioni pre-impostate tra cuiscegliere, pur lasciando l'utente libero di proporre una propria scelta.

ATTENZIONE

Poiché il controllo `datalist` non si distingue da quello di un campo di testo, siscopre 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`, 71

`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 e 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 dispazi 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 isuoi 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 disupporto 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 disupporto 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).

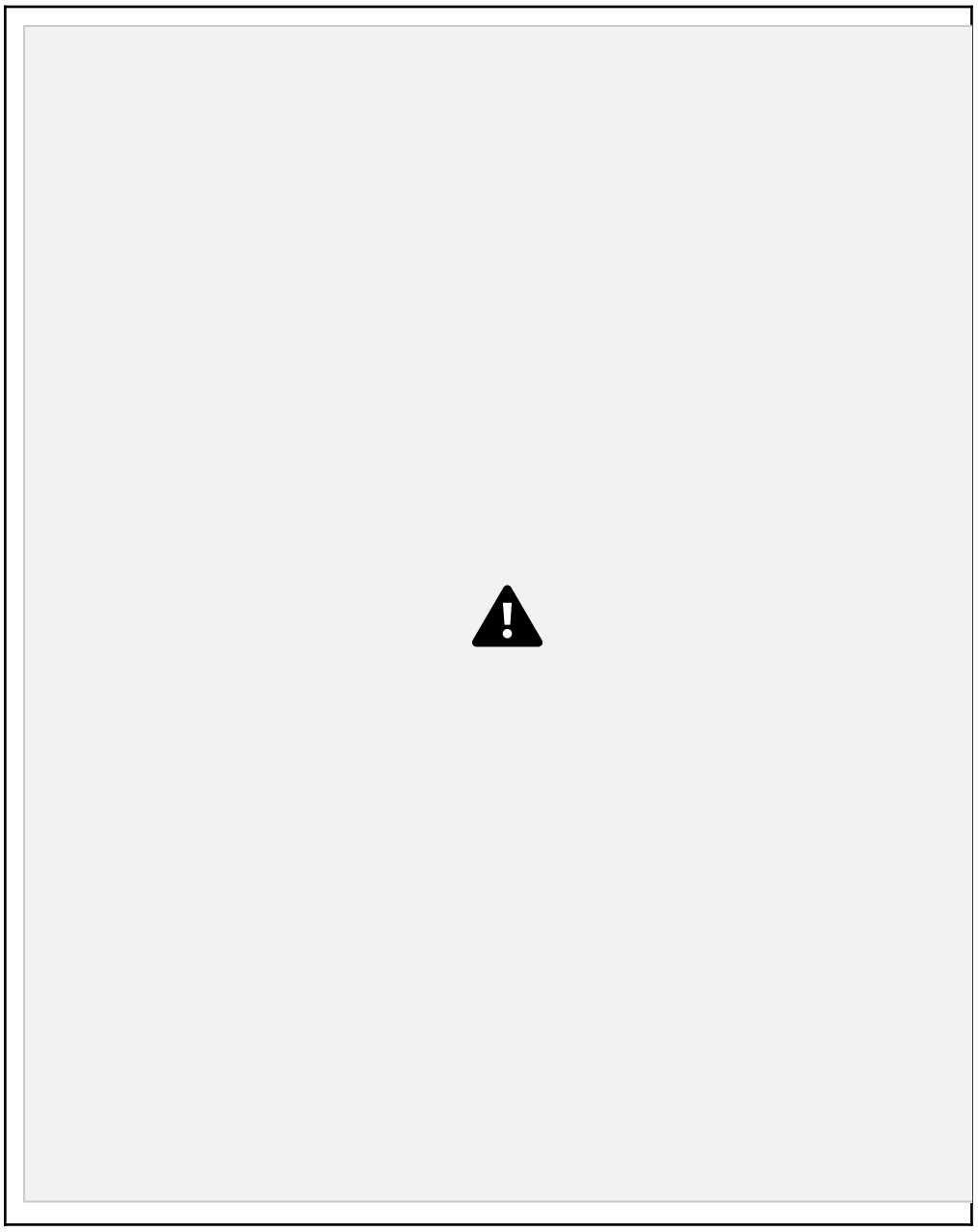

FIGURA 4.11 Il validatore HTML 4.01 segnala la presenza di un controllo del form in una posizione non autorizzata.

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).

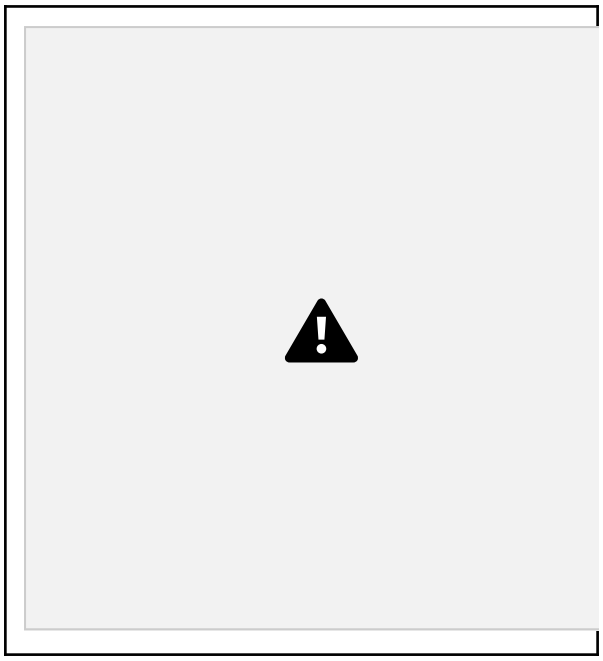

FIGURA 4.12 Il focus è restituito al campo in cui è stato violato il pattern dichiarato.

Un aiuto agli screen reader

Per consentire alle tecnologie assistive disvolgere 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.

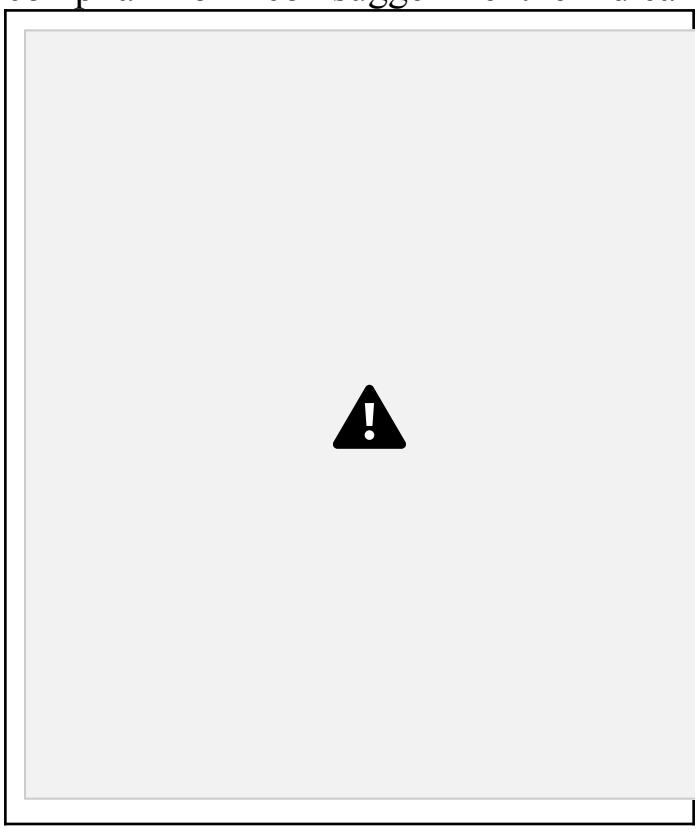

FIGURA 4.13 Il placeholder secondo Chrome.

La giusta lunghezza

È bene fare uno sforzo disintesi 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 marcatore:

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 ammissibile è, a oggi, `RSA` (acronimo ricavato dai nomi dei tre inventori dell'algoritmo: Rivest, Shamir e Adleman).

<output>

Questo marcatore 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>`.