Capitolo 1

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

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

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

Pragmatismo: questa è una delle parole chiave che hanno guidato lo sviluppo della nuova versione del linguaggio. Ciò ha significato prendere atto del potere discrezionale di cui godono i principali attori sul mercato dei browser nel decidere che cosa implementare e che cosa eliminare. Il tutto, partendo dalla constatazione che "non ha senso tradurre in specifica qualcosa che non può essere utilizzato perché una parte significativa dei visitatori non ne disporrà mai.

In questo capitolo

Meglio dedicarsi a tradurre in specifica cose che tutti implementeranno effettivamente" e sempre allo scopo di "rendere la specifica rispondente alla realtà" (`http://lists.whatwg.org/htdig.cgi/whatwg-whatwg.org/2010-June/026897.html`).
Un secondo principio cardine è stato il supporto diretto di tutte quelle soluzioni in cui JavaScript ha dimostrato di saper svolgere con efficacia il proprio compito.
Esempi di questa scelta sono evidenti nelle nuove funzionalità aggiunte ai form; per esempio, gli attributi `placeholder`, `autofocus` e `required` nascono proprio dalla consapevolezza che il diffuso impiego di JavaScript rivelava una lacuna delle precedenti versioni di HTML. Su questo tema si dirà di più nel Capitolo 4.
Un terzo pilastro della specifica è quello della *semantica*, tema ricorrente in diversi capitoli di questo manuale. HTML5 pone le basi per una migliore definizione dei vari componenti di una pagina web. I tag e gli attributi HTML vengono impiegati sempre più per fornire informazioni aggiuntive sui numerosi dati che popolano ogni pagina web di cui curiamo la pubblicazione. Una maggiore sensibilità verso questo aspetto significa aprire le porte a motori di ricerca più efficienti: si pensi al modo in cui già da più di un anno Google usa RDFa, microformat e microdata (formati di definizione e marcatura che definiscono uno standard per lo scambio di dati, in modo automatico, tra macchine) per offrire un set arricchito di informazioni partendo dai dati strutturati contenuti nelle pagine web create da ciascuno di noi. La semantica è anche la via attraverso la quale "liberare i dati" rendendone possibile la fruizione non solo a individui, ma anche direttamente ad altre macchine; in questo modo i dati possono essere rielaborati per assolvere funzioni diverse da quelle per cui originariamente erano stati resi disponibili. Infine, una più accurata descrizione del significato di un documento significa anche maggiore accessibilità per mezzo di tecnologie di supporto che, sfruttando una più approfondita conoscenza delle pagine web, possono svolgere meglio la loro funzione di supporto alla navigazione.
Uno degli scopi di questo manuale è di documentare che cosa si può fare già oggi con HTML5.
Per guidarvi lungo questo percorso troverete due utili strumenti, illustrati di seguito.

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

Una panoramica sulle differenti piattaforme web

Per testare accuratamente il comportamento e la visualizzazione delle pagine web sviluppate anche sulle più improbabili combinazioni di browser e sistemi operativi, il modo migliore consiste nel rivolgersi a uno dei diversi servizi promossi in Rete. Ne esistono sia gratuiti, come browsershots.org, 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 di sollecitare il prezioso riscontro degli utenti. Si potranno così ottenere informazioni per migliorare la fruibilità dei propri contenuti e al contempo fidelizzare i visitatori; è quel che si dice "saper fare di necessità virtù".

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

Modernizr vs soluzioni fai da te

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

Come ti scopro il browser!

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

Listato 1.1 Un esempio di stringa User Agent

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

Per una panoramica degli elementi che la compongono si può osservare la Figura 1.1.

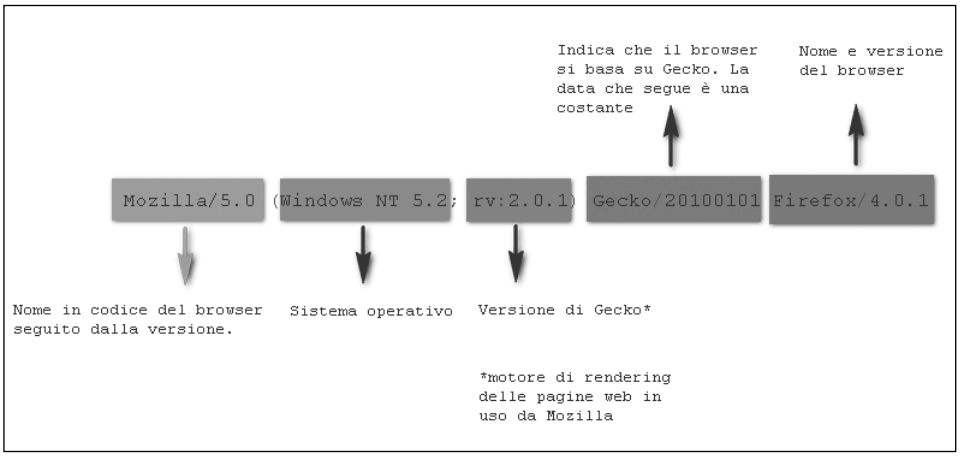

Figura 1.1 La stringa User Agent: un condensato di informazioni.

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.

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.

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.

Più browser sul mercato, più screenshot da browser diversi

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

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

La proprietà booleana `video` dell'oggetto `Modernizr` restituirà `true` se l'elemento `<video>` è supportato dal browser, `false` in caso contrario. Nel Capitolo 4, relativo alle potenzialità multimediali di HTML5, vedremo come utilizzare questa libreria anche per altre necessità.
Va da sé che questo approccio semplifica molto il lavoro dell'autore di pagine web, il quale dovrà, in questo caso, limitarsi a controllare gli aggiornamenti di una sola libreria

JavaScript, ed eventualmente intervenire con soluzioni *ad hoc* solo in caso di bachi.
Nel resto del presente manuale si utilizzerà questa libreria piuttosto che soluzioni di scripting "fai da te". A ogni modo, se nell'esempio sopra riportato si fosse voluto fare a meno di librerie esterne, si sarebbe potuto scrivere:

Listato 1.3 Script di identificazione del supporto all'elemento <video> (versione pedante)

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

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

    return bool;
  }
</script>
```

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

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

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

Indipendentemente dallo stile adottato, in questo caso notiamo come si tenda a verificare il supporto dell'elemento `video` leggendo il valore restituito da un suo metodo, `canPlayType`.
Questa è una delle quattro opzioni attraverso le quali valutare il supporto di una data funzionalità. Altre strategie verranno illustrate nel prosieguo del manuale.
Se desiderate stare alla larga da questa complessità, lavorando a un livello di astrazione più alto, e sviluppando al contempo un codice più sintetico e leggibile, oggi Modernizr (`http://www.modernizr.com/`) è la soluzione migliore.

Farsi un'idea delle funzionalità supportate

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

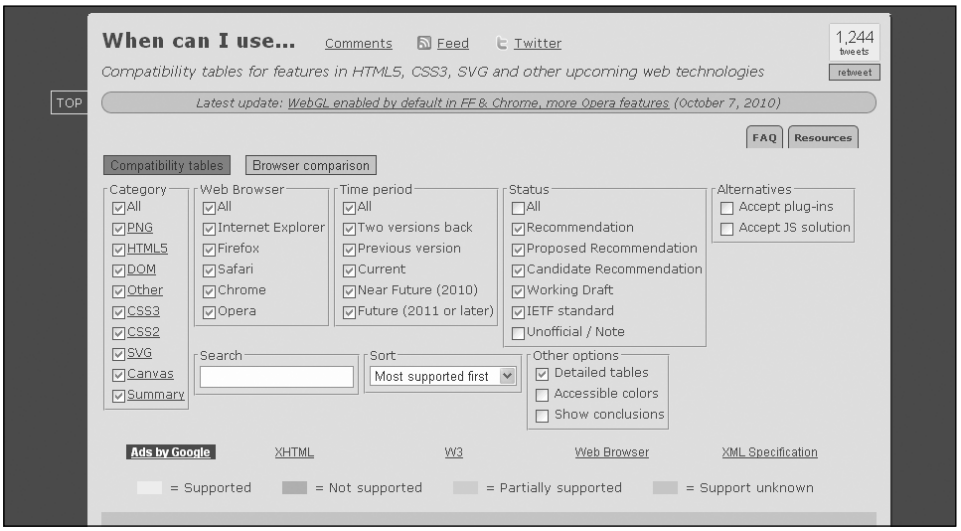

Figura 1.4 Guida pratica di compatibilità a HTML5, CSS3 e altre tecnologie legate al Web.

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).

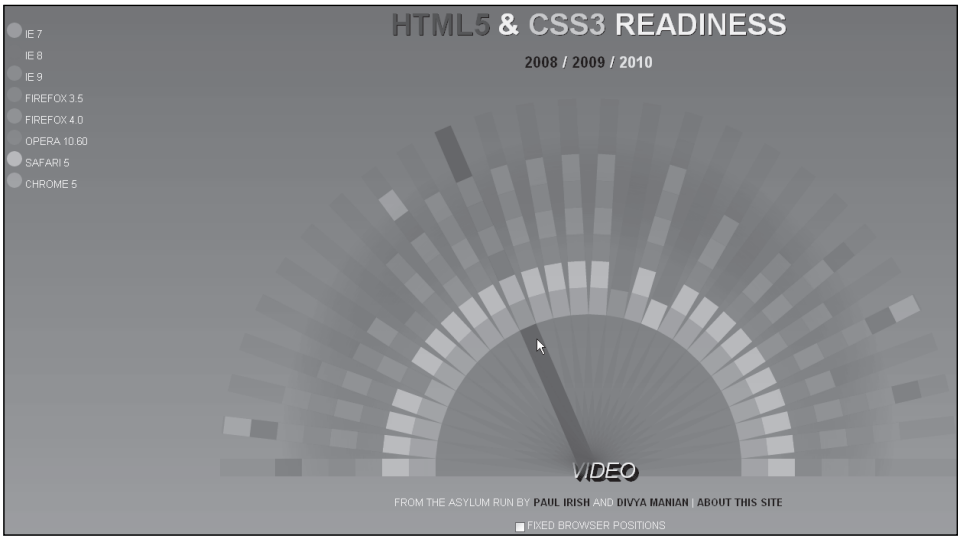

Figura 1.5 Un'accattivante rappresentazione grafica del livello di supporto rappresentato in base ai browser più diffusi e all'anno di riferimento.

Librerie JavaScript

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

1. Testare il supporto delle nuove funzionalità: è il caso della libreria Modernizr, che non interviene su ciò che un dato browser è in grado di fare, limitandosi a verificarne il riconoscimento di specifiche funzioni.
2. Sopperire alla mancata implementazione di date funzionalità, "abilitando" il browser all'uso di tecnologie per cui altrimenti non sarebbe pronto: rientra in questa categoria la libreria *excanvas*, che apre l'elemento `<canvas>` (cui si farà cenno nel Capitolo 6) al mondo dei browser di casa Microsoft, almeno nelle versioni precedenti alla 9.
3. Sperimentazione e applicazioni artistiche: sono queste le librerie più fantasiose che portano agli estremi una data tecnologia per enfatizzarne le potenzialità o solo per dimostrare come l'arte possa sposarsi anche con il Web. Un esempio in questo senso ci è offerto dalla libreria *close-pixelate* di David De Sandro.

La promessa di queste soluzioni sembra allettante: lasciare gli autori liberi di utilizzare subito il nuovo linguaggio, senza doversi preoccupare di sottoporre il codice a revisione quando i visitatori decideranno di aggiornare il proprio browser.
Proponiamo in Tabella 1.1 un elenco di alcune librerie che verranno discusse in questo manuale.

Tabella 1.1 Alcune librerie citate nel manuale

Libreria	Url	Descrizione
modernizr	`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 i sottotitoli 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 i superpoteri, ha voluto dimostrare come sia possibile creare un sistema di persistenza dei dati realmente efficace. I cookie generati in questo modo saranno salvati sulla macchina dell'utente in punti diversi utilizzando svariate tecniche. Se uno o più file dovessero essere rimossi, evercookie utilizzerebbe una delle altre copie ancora presenti per replicarsi nuovamente.

Riferimenti alle risorse citate e altri link utili

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

Conclusioni

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

Cicli di rilascio

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

I successivi capitoli, a partire dal prossimo, dedicato alla semantica dei documenti web, daranno un'idea più precisa del perché di tanto entusiasmo attorno a questo linguaggio. HTML5 rende possibile lo sviluppo di applicazioni web in modo più semplice e robusto di quanto non fosse possibile concepire anche solo fino a poco tempo fa.
Se nei prossimi anni assisteremo a una massiccia diffusione di applicazioni basate sulla specifica HTML5 e delle tecnologie web a essa legate, il Web come lo conosciamo oggi ci sembrerà preistoria.