

# CSS

Marco Squarcina

10 dicembre 2008



## Sommario

L'approfondimento riguarda i fogli di stile a cascata (da *Cascading Style Sheets*, CSS), un linguaggio utilizzato principalmente sul web per descrivere l'aspetto di documenti strutturati come HTML e XML. Vengono analizzate le motivazioni storiche che hanno portato all'introduzione dei CSS, il design, i problemi e la loro applicazione reale. Gran parte del materiale utilizzato per la stesura di questo articolo è stato ricavato dallo studio della tesi di PhD di Håkon Wium Lie, autore della bozza originale dei CHSS (poi rinominati CSS), ex-membro W3C e attuale CTO di Opera software; in mancanza di riferimenti espliciti, si presume che le informazioni presentate siano tratte dalla tesi precedentemente citata.

## Indice

<b>1</b>	<b>Cenni storici</b>	<b>3</b>
1.1	Struttura contro presentazione . . . . .	3
1.2	Livelli di astrazione . . . . .	4
1.3	Fogli di stile . . . . .	5
1.4	CSS . . . . .	6
<b>2</b>	<b>Design</b>	<b>7</b>
2.1	Sintassi . . . . .	7
2.2	Selettori . . . . .	8
2.3	Proprietà . . . . .	8
2.4	Valori e unità . . . . .	10
2.5	Trasmissione dei Valori . . . . .	10
2.5.1	Cascata . . . . .	11
2.5.2	Ereditarietà . . . . .	12
2.5.3	Valori iniziali . . . . .	12
2.6	Processing Model . . . . .	12
2.6.1	Modello di formattazione visuale . . . . .	13
2.7	Sistemi di collegamento . . . . .	13
2.8	Contenuto generato . . . . .	13
2.9	Media non visuali . . . . .	14
<b>3</b>	<b>Problemi</b>	<b>15</b>
3.1	Errori nelle specifiche . . . . .	15
3.2	Problemi nelle specifiche . . . . .	15
3.2.1	Mancanza di funzionalità . . . . .	15
3.2.2	Funzionalità in eccesso . . . . .	16
3.3	Problemi di design . . . . .	17
3.4	Problemi nelle implementazioni . . . . .	17
<b>4</b>	<b>Case study</b>	<b>19</b>
4.1	Accessi e statistiche pre-redesign . . . . .	19
4.2	Motivazioni . . . . .	19
4.3	Realizzazione . . . . .	19
4.4	Vantaggi . . . . .	20
4.5	Limiti . . . . .	21
<b>5</b>	<b>Conclusioni</b>	<b>22</b>

## 1 Cenni storici

All'incirca nel 1990, Tim Berners-Lee sviluppò tre specifiche che formarono la base del progetto World Wide Web: l'*HyperText Markup Language* (HTML) fu concepito come un formato di documento per il Web, gli *Universal Resource Locators* (URL) furono aggiunti per rappresentare i collegamenti tra i documenti e l'*HyperText Transfer Protocol* (HTTP) fu sviluppato per trasferire i documenti tra le macchine su Internet[5]. Sia le specifiche che le implementazioni furono rese liberamente disponibili dal CERN<sup>1</sup>.

Con l'uscita del browser Mosaic del *National Center for Supercomputing Applications* (NCSA) nel 1993[1], venne fornito agli utenti un modo comodo ed attraente per navigare un insieme crescente di documenti collegati tra loro. Il maggior numero di utenti favorì la crescita del numero degli autori ed il volume dei contenuti sul web aumentò rapidamente.

### 1.1 Struttura contro presentazione

Inizialmente l'HTML era un semplice formato di documento strutturato con dei *tag* di marcatura usati per definire la funzione di determinate parti testuali. Gli elementi erano logici piuttosto che presentazionali. L'HTML avrebbe potuto definire una frase come titolo, ma non avrebbe dato alcuna informazione sulla sua rappresentazione. Determinare il font, il colore e la dimensione era compito lasciato al browser.

Al di fuori dell'ambito scientifico, gli autori iniziarono presto a lamentarsi del fatto di non poter influire tramite l'HTML sull'aspetto delle loro pagine. Il seguente messaggio[2], inviato alla mailing list *www-talk*<sup>2</sup> all'inizio del 1994 da Marc Andreessen, uno dei programmatori del browser NCSA Mosaic, fornisce una fotografia dei rapporti fra autori e sviluppatori in quel periodo:

*In fact, it has been a constant source of delight for me over the past year to get to continually tell hordes (literally) of people who want to – strap yourselves in, here it comes – control what their documents look like in ways that would be trivial in TeX, Microsoft Word, and every other common text processing environment: ‘Sorry, you’re screwed.’*

Lo stesso Andreessen divenne successivamente co-fondatore di Netscape, browser rilasciato nell'ottobre del 1994[3] che rese disponibile il supporto ai primi tag HTML di tipo presentazionale. . .

Con l'aggiunta di questo tipo di tag, l'HTML si trasformò da linguaggio di marcatura astratto e strutturato dove gli autori definivano le differenti regole logiche del testo (paragrafi, intestazioni, elenchi, etc.) in un linguaggio di presentazione dove l'enfasi viene posta sulla forma finale dei documenti (font, colori e layout).

La trasformazione in linguaggio presentazionale iniziò a minacciare l'indipendenza dal dispositivo, l'accessibilità e il riutilizzo del contenuto: si pensi, per esempio, a come potrebbe venire rappresentato un testo lampeggiante (tag BLINK) da un sistema di sintesi vocale o da un display braille. Allo stesso tempo,

---

<sup>1</sup>European Organization for Nuclear Research

<sup>2</sup>Luogo di incontro elettronico della prima comunità web

gli utenti iniziarono a perdere sempre più controllo sul processo di formattazione visto che i browser ricevevano i documenti con delle regole di presentazione che determinavano la loro forma finale.

## 1.2 Livelli di astrazione

Prima di affrontare i fogli di stile, ritengo sia opportuno introdurre il concetto di scala d'astrazione. Un esempio di cosa si intenda con questo termine viene fornito dal campo delle reti di computer: nel 1983, la International Standards Organization (ISO) sviluppò un modello di network chiamato Open Systems Interconnection (OSI) Reference Model che definiva una struttura per le comunicazioni tra computer. L'ISO/OSI Reference Model è composto da sette strati<sup>3</sup>, ciascuno dei quali ha un diverso livello di astrazione.

Riguardo ai formati di documento, l'altezza sulla scala determina la complessità della formattazione del documento all'atto della presentazione. Caratteristiche tipiche dei formati ad alto livello di astrazione sono:

- L'informazione ha bisogno di elaborazione per essere presentata. Per esempio, per rendere visivamente un documento HTML le parole devono essere divise in righe, i font devono essere selezionati, e i caratteri devono essere convertiti in glifi rasterizzati.
- L'informazione può essere elaborata e presentata in molti modi diversi. Presentare un documento visivamente è solo una tra diverse possibilità; le altre comprendono la resa acustica e il braille.
- L'informazione è rappresentata in modo compatto. Rappresentare un carattere con una codifica a otto bit è più compatto che rappresentare un'immagine dello stesso carattere.[18]

I documenti scritti in formati che si trovano in basso sulla scala d'astrazione, invece, richiedono meno elaborazione per essere presentati, sono dotati di minore flessibilità nella presentazione e sono meno compatti.

Specificatamente al contesto dei documenti Web, Håkon Wium Lie, autore della bozza che diede origine ai CSS[17], membro del W3C<sup>4</sup> e attualmente CTO<sup>5</sup> di Opera Software, individua nella sua tesi di PhD i seguenti gradini nella scala d'astrazione:

- *Is the text human-readable? That is, if the document is presented to a human reader, will he/she be able to read the document?*
- *Is the text machine-readable? That is, does the format have a notion of numbered characters, or does it represent text as images – in which case the text is not available.*
- *Is the logical order of text preserved? That is, do documents written in the format have a notion of the logical reading order of the content?*

---

<sup>3</sup>fisico, collegamento dati, rete, trasporto, sessione, presentazione e applicazione.

<sup>4</sup>il *World Wide Web Consortium* (W3C) è la principale organizzazione internazionale per gli standard del *World Wide Web*. Fu fondata nel 1994 da Tim Berners-Lee dopo aver lasciato il CERN e divenne pienamente operativa nel 1995 includendo membri appartenenti ad aziende, organizzazioni nonprofit, università ed enti governativi.

<sup>5</sup>Chief technical officer

- *Is the document scalable? That is, can the document be zoomed in without introducing visible artefacts?*
- *Is reflow possible? That is, can text be reflowed into lines, columns and pages of different dimensions?*
- *Can the roles of the various text elements be represented? For example, can the author mark part of the text as a headline, a paragraph, or perhaps as the name of a variable in a computer program? Being able to distinguish between these roles is important. When making documents available in braille, for example, some text should be contracted (e.g. headlines), while other text should not (e.g. variable names).*
- *Is the format device-independent? That is, can documents written in the format be rendered into many different devices (e.g. printers, screens, braille printers, and text synthesizers) or are documents intended for a single type of device?*
- *Does the format contain application-specific semantics? HTML is a general document format that does not attempt to describe semantics from more specialized fields, e.g. mathematics and chemistry, and therefore does not contain application-specific semantics. Formats that contain application-specific semantics tend to be higher on the ladder of abstraction.*

### 1.3 Fogli di stile

I fogli di stile furono proposti per fermare il passo indietro nella scala d'astrazione dovuto all'evoluzione dell'HTML da linguaggio strutturale a linguaggio presentazionale. Il termine "foglio di stile" si riferisce ad un insieme di regole che esprimono come presentare un documento, ossia associano proprietà stilistiche e valori ad elementi strutturali. Generalmente non hanno contenuto, sono collegabili dai documenti e sono riutilizzabili[18]. Essendo direttamente responsabili della formattazione di un documento di testo, si capisce quanto sia importante il concetto dei livelli d'astrazione applicato ai fogli di stile.

I vantaggi derivanti dal loro utilizzo sul Web sono molteplici: un foglio di stile ben sviluppato fornirebbe agli autori un vocabolario stilistico più ricco rispetto alla possibile evoluzione dell'HTML, l'HTML stesso rimarrebbe un linguaggio di marcatura strutturato che funziona su un'ampia gamma di dispositivi e il riutilizzo dei contenuti verrebbe estremamente agevolato.

Per questi motivi, molti sviluppatori concordarono sulla convenienza di applicare tale sistema alle pagine Web. Tuttavia, non c'era un punto di vista comune sul linguaggio da adottare: alcuni ritenevano fosse più opportuno utilizzarne uno già esistente, concepito in origine per la pubblicazione cartacea, mentre altri propendevano per la creazione di un linguaggio di stile completamente nuovo.

Durante il 1993 furono proposti vari linguaggi, ma nessuno di questi venne considerato sufficientemente stabile per essere implementato dagli sviluppatori di browser e usato dagli autori.

## 1.4 CSS

Tre giorni prima che Netscape annunciasse il suo nuovo browser[1], Håkon Wium Lie pubblicò la prima proposta sui CSS[17]. Oltre a descrivere font, colori e layout di documenti (funzionalità già descritte in precedenti proposte) i CSS introducevano il concetto di *cascata*, una nuova funzionalità che teneva conto delle differenze di pubblicazione imposte dal Web, in modo da rendere possibile sia agli autori che agli utenti influire sulla presentazione di un documento.

L'ambizione dei CSS era quella di fornire uno strumento grazie al quale gli autori riuscissero ad occuparsi della formattazione dei propri documenti senza sentire il bisogno di ricorrere ad HTML presentazionale, mentre gli utenti potessero decidere se accettare la presentazione proposta dall'autore o utilizzarne una propria. Inoltre il browser avrebbe dovuto fornire un foglio di stile di default da applicare agli elementi HTML. I CSS individuavano quindi tre possibili sorgenti per i fogli di stile: autori, utenti e browser. La capacità di combinare i fogli di stile forniti da queste tre fonti, al fine di visualizzare il documento nella forma finale, è detto *cascata*[18].

Alla proposta seguirono vari messaggi[6][4][21] e la bozza fu ulteriormente sviluppata. Durante il 1995, furono pubblicate circa otto revisioni. L'ultima (dicembre 1995) fu dichiarata stabile e i produttori di browser furono "incoraggiati a usarla come base per le implementazioni"[22].

Le specifiche CSS1 [7] divennero una Raccomandazione W3C nel dicembre 1996 e due anni più tardi furono i CSS2 [9] a diventare una Raccomandazione W3C.

Ad oggi, tutti i maggiori browser supportano i CSS e la maggioranza delle pagine web li usa. Una Revisione del livello 2 dei CSS (CSS2.1) è in fase di *Candidate Recommendation* [11]. Il livello 3 dei CSS (CSS3) è in fase di sviluppo dal 1998 e a differenza dei precedenti sarà *modulare* essendo composto da una serie di Raccomandazioni separate<sup>6</sup>.

---

<sup>6</sup>Per informazioni riguardo alla modularizzazione delle specifiche CSS3, si faccia riferimento a <http://www.w3.org/TR/css3-roadmap/>

## 2 Design

Nella precedente sezione sono state trattate le origini storiche dei fogli di stile e sono stati introdotti concetti fondamentali come linguaggio di presentazione e livelli di astrazione. Viene ora fornita una introduzione al design dei CSS.

### 2.1 Sintassi

La sintassi dei CSS è abbastanza semplice. Prendiamo come esempio la seguente regola:

```
p {
    font-size: 3em
}
```

La regola appena presentata è composta da un *selettore*<sup>7</sup> (`p`) e da una *dichiarazione* (`font-size: 3em`). A sua volta la dichiarazione è formata da *proprietà* (`font-size`) e *valore* (`3em`).

È possibile raggruppare più dichiarazioni all'interno di un unico blocco:

```
p {
    font-size: 3em;
    color: blue;
}
```

Il punto e virgola viene usato per separare le dichiarazioni (tranne l'ultima, dove è opzionale).

Per assegnare il contenuto di un blocco a più selettori, invece, è sufficiente dividerli con una virgola:

```
h3, h4 {
    color: green;
    margin: 2em;
}
```

`margin` è un esempio di *proprietà abbreviata*, un modo per definire il valore di più *proprietà individuali* in una sola dichiarazione. Utilizzando esclusivamente proprietà individuali, il codice CSS precedentemente riportato equivale a:

```
h3, h4 {
    color: green;
    margin-top: 2em;
    margin-right: 2em;
    margin-bottom: 2em;
    margin-left: 2em;
}
```

Alcune proprietà CSS, come `font-family`, accettano elenchi di valori separati da virgole e settano il valore in funzione alla disponibilità della risorsa indicata dal primo all'ultimo elemento della lista.

Gran parte della logica nei CSS viene espressa tramite selettori:

<sup>7</sup>*search pattern* che identifica la parte di documento a cui viene applicata la relativa dichiarazione

```
div.header p:first-line {
    text-transform: capitalize;
}
```

In questo esempio viene specificato che la prima lettera di ogni parola della prima riga di tutti gli elementi `p` all'interno di `div` di classe `header`, deve essere trasformata in un carattere maiuscolo.

Al fine di permettere a futuri livelli dei CSS di includere nuove funzionalità, assicurando allo stesso tempo retrocompatibilità con *parser*<sup>8</sup> precedenti, i nuovi selettori, proprietà e valori devono venire riconosciuti ed ignorati dalla precedente implementazione.

Determinati costrutti sintattici avanzati sono implementati tramite *at-keyword*, regole precedute dal simbolo `@`. CSS1 usava una sola *at-keyword* per importare un foglio di stile all'interno di un altro, mentre CSS2 ne introdusse quattro per descrivere il set di caratteri usato nel foglio di stile, le risorse di font scaricabili, i media impaginati e i fogli di stile dipendenti da un media specifico.

## 2.2 Selettori

Nei CSS1 furono introdotti selettori *semplici* e *contestuali*. Mentre i primi identificano un elemento in base al suo tipo o ai suoi attributi, i selettori contestuali (composti da diversi selettori semplici) effettuano la selezione in base alla posizione dell'elemento nella struttura del documento.

Al fine di estendere le possibilità di “decorazione” dei documenti e aumentare la separazione fra contenuto e stile, vennero introdotti gli *pseudo-elementi*, selettori che non identificano elementi realmente presenti nel codice sorgente, ma che corrispondono ad oggetti di visualizzazione. Nei CSS1 sono presenti due pseudo-elementi: `first-letter` e `first-line` che si riferiscono rispettivamente alla prima lettera e alla prima riga di un elemento, in funzione di come appare sullo schermo.

Per visualizzare uno stesso elemento con uno stile differente in base ad una determinata informazione esterna (come per esempio un'azione dell'utente), furono aggiunte le *pseudo-classi*. Le più famose sono sicuramente `link`, `visited` e `active`, introdotte nei CSS1 ed estremamente usate per decorare i link ipertestuali.

Una lista dei selettori presenti nei CSS1 è disponibile nella tabella 1, mentre nella tabella 2 sono descritti quelli introdotti nei CSS2. È importante notare come i selettori aggiunti al secondo livello dei fogli di stile conferiscano espressività tale da rendere i CSS applicabili a linguaggi differenti dall'HTML.

## 2.3 Proprietà

Mentre i CSS1 descrivono delle proprietà di base per la formattazione visuale, i CSS2 ampliano tale insieme includendo il supporto alla stampa e alle presentazioni acustiche.

Per la lista completa delle proprietà definite dal secondo livello dei fogli di stile, si faccia riferimento all'Appendice F<sup>9</sup> della raccomandazione W3C [9].

<sup>8</sup>algoritmo di riconoscimento sintattico di un linguaggio

<sup>9</sup><http://www.w3.org/TR/CSS2/propidx.html>

Tabella 1: Lista dei selettori presenti nei CSS1

Pattern	Significato
E	Seleziona ogni elemento E (ossia un elemento di tipo E).
E F	Questo selettore contestuale seleziona ogni elemento F che sia discendente di un elemento E.
E:link E:visited	Seleziona l'elemento E se E è l'ancora di un ipercollegamento il cui target non è stato ancora visitato (:link) o è stato già visitato (:visited).
E:active E:hover E:focus	Seleziona E durante determinate azioni da parte dell'utente.
DIV.warning	Lo stesso che DIV[class =warning] nei CSS2.
E#myid	Seleziona ogni elemento E con ID uguale a myid.

Tabella 2: Lista dei selettori presenti nei CSS2

Pattern	Significato
*	Il selettore universale che seleziona ogni elemento.
E > F	Seleziona ogni elemento F figlio di un elemento E.
E:first-child	Seleziona l'elemento E quando E è il primo figlio del suo genitore.
E:lang(c)	Seleziona l'elemento di tipo E se esso è in una lingua (umana) c (il linguaggio del documento specifica come viene determinata la lingua).
E + F	Seleziona ogni elemento F immediatamente preceduto da un elemento E.
E[foo]	Seleziona ogni elemento E che abbia l'attributo foo impostato (qualunque sia il valore).
E[foo="warning"]	Seleziona ogni elemento E il cui valore dell'attributo foo è esattamente uguale a warning.
E[foo~="warning"]	Seleziona ogni elemento E il cui valore dell'attributo foo è un elenco di valori separati da uno spazio, uno dei quali è esattamente uguale a warning.
E[lang "en"]	Seleziona ogni elemento E il cui attributo lang ha un elenco di valori separati da trattino che cominciano (da sinistra) con en.

Come già esposto nella sottosezione 2.1 a pagina 7, alcune proprietà godono di una sintassi abbreviata per definire diversi valori in una sola dichiarazione con il vantaggio di ridurre la dimensione dei fogli di stile e aumentarne la leggibilità.

## 2.4 Valori e unità

I CSS offrono sei tipi base di valori:

**keyword:** tutte le proprietà accettano una o più keyword. Nei CSS2 `inherit` viene accettata da tutte le proprietà, mentre le keyword maggiormente utilizzate sono `normal`, `none`, `auto`, `left`, `bottom`, `right` e `top`;

**stringhe:** sono racchiuse fra singoli o doppi apici e vengono utilizzate quando il *namespace* non è limitato;

**numeri:** possono essere sia interi che reali. Alcune proprietà accettano solo valori interi (come nel caso di `z-index`), mentre altre sia interi che reali (`line-height` per esempio);

**numeri con unità:** vengono usati soprattutto per esprimere lunghezze, ma possono rappresentare anche angoli (es. `180deg`), intervalli di tempo (es. `40ms`) e, per quanto riguarda le proprietà aurali, frequenze (es. `9khz`);

**percentuali:** (ad esempio `55%`) sono sempre relative ad un altro valore definito dalla proprietà stessa, come la larghezza dell'antenato di blocco più vicino o la dimensione del font dell'elemento genitore;

**funzioni:** alcune proprietà avanzate (come `content`) accettano una notazione funzionale quando il valore si riferisce ad un oggetto arbitrario o non è esprimibile tramite una stringa.

Per quanto riguarda le unità di misura, i CSS distinguono fra assolute e relative. Si veda la Tabella 3 per la lista completa.

Tabella 3: Lista delle unità di misura presenti nei CSS2

Unità di misura assolute	
Unità	Significato
in	pollici
cm	centimetri
mm	millimetri
pt	punti tipografici (corrispondenti a 1/72 di pollice)
pc	picas (corrispondenti 12 punti tipografici)
Unità di misura relative	
Unità	Significato
em	dimensione del font dell'elemento corrente
ex	altezza del carattere "x" rappresentato con il font dell'elemento corrente
px	pixel

## 2.5 Trasmissione dei Valori

I CSS dispongono di tre meccanismi per la propagazione dei valori: *cascata*, *ereditarietà* e *valori iniziali*. Il sistema della cascata è il più forte, poiché se

produce un valore, questo verrà usato. Segue quello dell'ereditarietà ed infine, se nessuno dei due meccanismi precedenti ha prodotto un valore, subentrerà il sistema dei valori iniziali.

### 2.5.1 Cascata

Quando si verifica un conflitto fra dichiarazioni di uno stesso elemento, il sistema di cascata deve scegliere la dichiarazione da utilizzare in base a tre fattori: la sorgente, la *specificità*<sup>10</sup>, e l'ordine di apparizione nel foglio di stile.

Come anticipato nella sezione 1.4, le sorgenti di un CSS possono essere l'autore, l'utente e il browser. Per impostazione predefinita, le dichiarazioni dell'autore sono più forti di quelle dell'utente, a loro volta più forti di quelle del browser. All'utente viene comunque data la possibilità di definire la presentazione finale di un documento poichè le dichiarazioni marcate come `!important` vincono su quelle dell'autore.

Se la sorgente non produce una dichiarazione vincente, si deve calcolare la specificità del selettore associato alle dichiarazioni. Si consideri il seguente esempio:

```
* { color: silver; }
li { color: red; }
ul li { color: blue; }
li.warning { color: green; }
```

Il primo selettore (`*`), detto *selettore universale*, è il più generico e seleziona tutti gli elementi del documento. Intuitivamente il secondo è più specifico del precedente poichè seleziona solo gli elementi `li`. Il terzo è un selettore contestuale che seleziona gli elementi `li` contenuti in un elemento `ul`, mentre il quarto seleziona gli elementi `li` con un attributo `warning`. Capire quale sia il selettore più forte fra gli ultimi due non è semplice come nel caso precedente.

I CSS definiscono quindi una regola per calcolare in modo preciso la specificità:

- sia  $A$  il numero di attributi ID nel selettore;
- sia  $B$  il numero degli altri attributi e pseudo-classi nel selettore;
- sia  $C$  il numero dei nomi di elemento nel selettore;
- si ignorino gli pseudo-elementi.

La concatenazione di  $A - B - C$  definisce la specificità dell'elemento.

Il selettore `ul li` riportato nell'esempio avrà quindi specificità 2 poichè:

$$(A = 0, B = 0, C = 2) \Rightarrow A - B - C \Rightarrow 002 = 2$$

`li.warning`, invece, avrà specificità 11:

$$(A = 0, B = 1, C = 1) \Rightarrow A - B - C \Rightarrow 011 = 11$$

Infine, se più dichiarazioni in conflitto hanno al stessa specificità, l'ordine in cui appaiono nei fogli di stile determina il vincitore (le successive vincono su quelle precedenti).

<sup>10</sup>misurazione di quanto sia esplicito un selettore

### 2.5.2 Ereditarietà

Tutte le proprietà definite dai CSS accettano la keyword `inherit` la quale specifica che il valore debba venire preso dal genitore. Se anche l'elemento radice ha la keyword `inherit`, vengono usati i valori iniziali.

I CSS distinguono tra valori specificati, calcolati ed effettivi. Mentre i valori specificati si trovano nei fogli di stile e i valori calcolati vengono elaborati nella misura possibile senza rappresentare il documento, i valori effettivi sono quelli realmente utilizzati per effettuare la *rendering*. In genere, è il valore calcolato di una proprietà ad essere ereditato, ma le proprietà possono specificare che altri tipi di valori debbano invece esserlo, come ad esempio `line-height` che eredita il valore specificato solo se questo è un numero.

### 2.5.3 Valori iniziali

Ogni proprietà CSS ha un valore iniziale che diviene valore risultante se la cascata e l'ereditarietà non producono alcun valore. Inoltre il valore iniziale può essere esplicitamente assegnato tramite la keyword `initial`, accettata da tutte le proprietà.

## 2.6 Processing Model

Il *Processing Model* definisce lo schema concettuale di come funziona il supporto ai CSS da parte degli *user agent*<sup>11</sup> (UA). In base a questo modello, l'elaborazione del sorgente di un documento avviene secondo i seguenti passi [12]:

1. Analisi del sorgente e creazione dell'albero degli elementi del documento.
2. Identificazione del tipo di media<sup>12</sup>.
3. Download dei fogli di stile per il media specificato associati al documento.
4. Annotazione di ciascun elemento dell'albero in modo da assegnare un singolo valore ad ogni proprietà, secondo il meccanismo descritto nella sezione 2.5. Una parte del calcolo dei valori dipende dall'algoritmo di formattazione appropriato al media specificato. Nel caso in cui il media sia `screen`, verrà applicato il modello di formattazione visuale (*visual formatting model*, VFM).
5. Generazione della struttura di formattazione a partire dall'albero con annotazioni.
6. Trasferimento della struttura di formattazione sul media indicato (rendering sullo schermo, stampa, sintetizzazione vocale, etc.).

Sebbene l'accesso non visuale ai documenti sia stato importante nello sviluppo dei CSS, le presentazioni visuali del testo sono state indubbiamente quelle di maggior impiego e la chiave del successo dei CSS.

---

<sup>11</sup>browser

<sup>12</sup>all, braille, embossed, handheld, print, projection, screen, speech, tty, tv

### 2.6.1 Modello di formattazione visuale

Il VFM descrive quindi come lo user agent debba processare l'albero degli elementi del documento per i media visuali. Secondo questo modello, ciascun elemento appartenente all'albero deve generare 0 o più *box* in base alle regole stabilite dal *box model* <sup>13</sup>. La rappresentazione grafica di queste box è definita in base a:

- dimensioni e tipo dei box;
- schema di posizionamento (*normal flow*, *float* e *absolute*);
- relazioni fra elementi all'interno dell'albero;
- informazioni esterne (come per esempio la dimensione della finestra di visualizzazione della pagina o la grandezza delle immagini).

Ad ogni modo, il modello di formattazione visuale non fornisce le specifiche per tutti i possibili aspetti riguardanti la formattazione, alcuni dei quali sono lasciati all'interpretazione dello user agent (come per esempio l'algoritmo per effettuare il letter-spacing).

## 2.7 Sistemi di collegamento

Nel 1996, quando i CSS1 divennero una Raccomandazione W3C [7], le specifiche HTML2 [15] non presentavano alcun sistema per collegare i documenti HTML ai fogli di stile. Sebbene la definizione del meccanismo di collegamento andasse oltre alla specifica dei CSS, gli elementi `link`, `style` e l'attributo `style` furono successivamente aggiunti ad HTML4 [16] nel 1997. Risale al 1999 invece la specifica W3C per associare i fogli di stile a documenti XML [25].

## 2.8 Contenuto generato

Con i CSS2 è possibile generare del contenuto prima e dopo un elemento all'interno di un documento definendo la proprietà `content` per gli pseudo elementi `:before` e `:after`. Il seguente esempio illustra come aggiungere la parola "Sottosezione:" prima di ogni elemento `h3`:

```
h3:before { content: "Sottosezione: "; }
```

La proprietà `content` accetta valori di tipo stringa, URL, keyword per definire le virgolette e contatori. Quest'ultimi sono inizializzati dalla proprietà `counter-reset` e incrementati/decrementati da `counter-increment`. Il seguente codice fornisce un esempio per la numerazione di liste ordinate annidate:

Sorgente CSS:

```
li { list-style: none; }
ol { counter-reset: item }
li:before {
    content: counter(item);
    counter-increment: item;
}
```

<sup>13</sup><http://www.w3.org/TR/CSS2/box.html>

Sorgente XHTML:

```
<ol>
  <li></li>
  <li>
    <ol>
      <li></li>
      <li></li>
      <li></li>
    </ol>
  </li>
</ol>
```

Risultato formattato:

```
1
2
  1
  2
  3
3
```

Come si può vedere dall'esempio riportato, la funzione `counter()` restituisce il valore del contatore `item`, associato al numero di elementi `li`, mantenendo lo stack dei contatori aperti.

## 2.9 Media non visuali

Uno dei benefici principali alla base dei fogli di stile è poter riproporre facilmente il contenuto per vari tipi di media. Oltre ai dispositivi visuali per la rappresentazione dei contenuti web, utenti con disabilità visive o uditive necessitano presentazioni su dispositivi acustici o schermi braille.

I CSS si sforzano di supportare contesti di formattazione multipli e i CSS2 hanno introdotto due caratteristiche fondamentali per supportare l'accesso multimodale ai contenuti web: ACSS (*Aural Cascading Style Sheets*) per definire come i documenti debbano essere resi su un dispositivo acustico e la possibilità di definire a quale tipo di dispositivo debba applicarsi una particolare regola di stile tramite l'attributo `media` (parte di HTML4 [16]).

## 3 Problemi

Dopo aver analizzato il design dei fogli di stile, rimangono da trattare i problemi relativi ai CSS. In questa sezione vengono affrontati i problemi che vanno da semplici errori nelle specifiche a problematiche strutturali ben più complesse, fino ai limiti posti dalle implementazioni dei CSS nei browser.

### 3.1 Errori nelle specifiche

Le specifiche CSS non sono esenti da errori. Questi vengono raccolti in documenti che accompagnano le specifiche stesse fino a quando non viene rilasciata una revisione con le opportune correzioni.

Nel caso dei CSS1, le specifiche vennero ripubblicate [8] nel giro di due anni, mentre nei CSS2 [10] furono rilevati problemi di carattere semantico che stanno portando alla pubblicazione di una nuova versione [11].

### 3.2 Problemi nelle specifiche

Mentre la correzione di un errore nella specifica risulta molto spesso un'azione triviale, l'aggiunta o la rimozione di determinate funzionalità o i problemi derivanti da un cattivo design, richiedono un dispendio di energie estremamente superiore.

Negli anni è emersa la necessità di introdurre nuove funzionalità, mentre altre, soprattutto per la mancanza di supporto da parte dei browser, non hanno quasi mai trovato un utilizzo.

#### 3.2.1 Mancanza di funzionalità

La lista seguente riporta degli esempi di carenze [18][24] percepite dai designer nel corso degli anni.

**contrasto di colore:** non è possibile assicurare un determinato contrasto tra il colore del testo e lo sfondo;

**dimensioni dei font:** non è possibile specificare la dimensione del font in funzione della larghezza dell'elemento;

**interlinea:** non è possibile impostare l'interlinea in funzione della grandezza dell'elemento;

**centratura verticale:** non è possibile centrare un elemento verticalmente in relazione allo schermo;

**layout multi-colonna:** non è possibile descrivere un layout multi-colonna in modo che il contenuto venga distribuito automaticamente da una colonna all'altra;

**forma degli elementi:** non è possibile specificare la forma degli elementi, poiché i CSS definiscono solo box rettangolari;

**background multipli:** non è possibile impostare più di una immagine di sfondo per elemento, costringendo i designer a creare degli ulteriori box per ottenere l'effetto desiderato.

Alcuni limiti emersi nei CSS1 furono superati con la pubblicazione della specifica per i CSS di secondo livello. Un procedimento simile sta portando alla definizione del terzo livello dei fogli di stile, in modo da rendere questo linguaggio ancora più potente ed espressivo.

### 3.2.2 Funzionalità in eccesso

Se da un lato i designer hanno percepito la necessità di introdurre nuove funzionalità nei fogli di stile, gli sviluppatori di browser hanno ritenuto troppo complesso implementarne altre. In molti casi alcune funzionalità non vennero praticamente mai utilizzate proprio a causa della mancanza di interoperabilità derivante da un supporto nullo o parziale degli user agent.

Vengono riportati di seguito alcuni esempi di funzionalità presenti nei CSS2 scarsamente utilizzate per il motivo sopracitato:

**text-shadow:** proprietà aggiunta per generare effetti d'ombra sul testo con lo scopo di scoraggiare gli autori ad utilizzare immagini al posto di elementi testuali. Fu considerata complessa da implementare e non dava la possibilità di rappresentare tutti gli effetti ottenibili tramite immagini.

**marks:** tramite rappresentazione grafica di croci e linee, fornisce una indicazione su dove i fogli debbano essere tagliati (dopo la stampa). La stessa operazione può venire svolta dall'utente usando la finestra di dialogo per la stampa fornita dall'applicazione.

**font scaricabili:** funzionalità introdotta per permettere di scaricare i font dal Web in modo da aumentare la ricchezza della presentazione. Non ebbe successo a causa dell'assenza di un formato universale per i font che fosse supportato da tutti i produttori e per la mancanza di una modalità di pagamento dei font stessi.

**font-size-adjust:** proprietà introdotta per preservare **x-height** quando una famiglia di font viene sostituita da un'altra, senza tenere conto della dimensione del font.

La mancanza di supporto a queste funzionalità portò gli sviluppatori W3C a rivedere la specifica stessa dei CSS2. Nella Candidate Recommendation per i CSS2.1 [11], infatti, le proprietà elencate precedentemente sono state rimosse, insieme alle seguenti [19]:

- font-stretch
- marker-offset
- page
- size
- molte altre riguardanti le presentazioni aurali...

### 3.3 Problemi di design

Se l'introduzione o la rimozione di determinate funzionalità è una questione complessa, ancora maggiore è una modifica del design dei fogli di stile.

Nel corso degli anni sono state avanzate molte critiche verso caratteristiche strutturali dei CSS; in questo paragrafo ne vengono presentate alcune:

**positività:** le regole CSS possono esprimere solo asserzioni positive, ma non esiste un modo per definire che valori particolari, o combinazioni di valori, non siano accettabili.

**marcatatura generica:** un designer può scrivere un foglio di stile per una pagina HTML usando gli elementi consentiti dal linguaggio nei selettori CSS. Qualora il documento utilizzi una marcatura generica, elementi e attributi saranno conosciuti soltanto dall'autore del documento stesso. In questo contesto, soltanto il selettore universale e i pseudo-selettori assumono un significato.

**antenati dei selettori:** tramite i selettori, non esiste un modo per selezionare il padre o un antenato di un elemento che soddisfi determinati requisiti<sup>14</sup>.

**ereditarietà:** non esiste un sistema che permetta ad un blocco di dichiarazioni di ereditare i valori da un altro.

**variabili:** non esistono variabili nei CSS. Questo comporta grosse difficoltà ai designer quando devono cambiare un valore ricorrente all'interno del foglio di stile.

### 3.4 Problemi nelle implementazioni

Sebbene i fogli di stile a cascata di primo livello siano stati completati nel 1996, lo standard non divenne realmente usabile fino al 2001 [20].

I browser esistenti fino al 1999, come *Microsoft Internet Explorer 3 e 4* e *Netscape-3 e 4*, implementavano interoperabilmente solo una piccola parte delle proprietà dei CSS. Jeffrey Zeldman descrisse nei seguenti termini lo stato del supporto ai fogli di stile nel 1998:

*If Netscape 4 ignored CSS rules applied to the <body> element and added random amounts of whitespace to every structural element on your page, and if IE4 got <body> right but bungled padding, what kind of CSS was safe to write? Some developers chose not to write CSS at all. Others wrote one style sheet to compensate for IE4's flaws and a different style sheet to compensate for the blunders of Netscape 4.*

La situazione cambiò gradualmente con la diminuzione dell'uso di Netscape in favore dell'aumento della diffusione di Explorer e il rapido miglioramento del supporto ai CSS in quest'ultimo.

---

<sup>14</sup>l'aggiunta di questa funzionalità, sebbene utile, andrebbe contro il principio di formattazione incrementale, quindi è estremamente poco probabile che venga implementata.

Nel 2000 Netscape rilasciò la versione 6 del suo browser, prima release ad utilizzare il nuovo *motore di rendering*<sup>15</sup> *Geko*. L'anno successivo Microsoft rese disponibile Internet Explorer 6, il primo browser che, per quanto incompleto e affetto da numerosi bug, ebbe un supporto usabile ai fogli di stile. Seguirono altri browser con un ottimo supporto ai CSS, tra i quali Opera, Mozilla e Safari.

Attualmente, il principale limite allo sviluppo dei fogli di stile a cascata sul Web è dovuto alla grandissima diffusione di Internet Explorer, il cui motore di rendering *Trident* viene considerato il peggiore tra i più usati [23]. A causa della posizione dominante che riveste IE nel mercato dei browser, si è definito fra gli autori un sottoinsieme di proprietà CSS utilizzabili al fine di creare presentazioni interoperabili e retrocompatibili.

---

<sup>15</sup>software che dato un contenuto scritto in un linguaggio di markup e le informazioni di formattazione relative, si occupa della rappresentazione della forma finale del documento su un dispositivo di output.

## 4 Case study

Nel Capitolo 1 sono state esposte le ragioni e i vantaggi relativi all'introduzione dei fogli di stile per i documenti Web, mentre nel Capitolo 3 è stata fornita una panoramica sui limiti e i problemi dei CSS. Resta quindi da analizzare quale sia l'impatto di questa tecnologia nelle applicazioni reali.

Nel 2003 ESPN<sup>16</sup>, televisione Americana via cavo dedicata allo sport, seguita da oltre 100 milioni di persone e diffusa in oltre 150 paesi nel mondo, decise di effettuare un redesign del suo sito internet passando da un layout tabellare all'utilizzo di un linguaggio di markup strutturale presentato tramite CSS.

L'attenzione dei developer Netscape fu catturata dal processo di redesign di ESPN.com al punto che Eric A. Meyer, al tempo manager Netscape, inviò una serie di domande via e-mail a Mike Davidson, direttore artistico della società. Le risposte che arrivarono furono estremamente dettagliate e vennero interamente pubblicate sotto forma di intervista [13][14]. Tale materiale costituisce il dominio informativo di questo capitolo.

### 4.1 Accessi e statistiche pre-redesign

Nel periodo 2002-2003, l'insieme di tutti i siti di proprietà di ESPN registrava in media 1,2 miliardi di pagine visitate mensilmente. Poco prima che venisse affrontato il redesign del sito, si calcolarono gli accessi in base ai browser utilizzati. Risultò che la percentuale di visite effettuate con browser *standards-compliant*<sup>17</sup> si attestava al 97,44%, mentre l'1,32% con Internet Explorer 4.x, l'1,7% con Netscape 4.x e il restante 0,05% non era identificabile.

### 4.2 Motivazioni

Mike Davidson indicò nella *forward compatibility* la motivazione principale che portò all'abbandono del vecchio layout tabellare. Con questo termine, il direttore artistico di ESPN.com intendeva la possibilità di estendere la presentazione dei contenuti al di fuori dei sistemi di visualizzazione tradizionali, considerati fino ai primi anni del nuovo millennio i Web browser grafici su desktop o laptop. Con l'avvento di *smartphone*, *tablet pc*, *digital lifestyle device* e altri apparecchi, il portale doveva prepararsi ad una visualizzazione ottimale da parte di una nuova classe di dispositivi.

Una home page da 100KB ricca di filmati Flash, immagini e tabelle non era adatta ad un dispositivo wireless dalle capacità limitate e con uno schermo ridotto. La separazione del contenuto dalla presentazione grafica, poteva permettere di definire differenti layout per ogni device che supportasse standard aperti.

### 4.3 Realizzazione

La struttura di ESPN.com era estremamente complessa in quanto differenti layout venivano impiegati per la visualizzazione di determinate sezioni. Affrontare

---

<sup>16</sup>acronimo di Entertainment and Sports Programming Network

<sup>17</sup>Mike Davidson definì *standards-compliant* i seguenti browser: Internet Explorer 5+, Mozilla, Opera 6+, Safari, Chimera e Konqueror

il redesign di tutto il sito era un'operazione irrealizzabile, quindi si optò per modificare la home page e proseguire poi con il lavoro sulle pagine correlate.

La struttura fu riscritta completamente: al termine del processo di redesign, nella pagina principale del sito non c'era più alcuna tag `font` e le uniche due tabelle presenti erano utilizzate per il banner e per presentare dei contenuti all'interno di una griglia. A causa della mancanza di interoperabilità da parte dei browser nella visualizzazione di `div float`, si scelse di utilizzare un posizionamento di tipo assoluto.

Furono create quattro modalità di visualizzazione per l'home page:

**regular:** layout utilizzato per il 90% del tempo caratterizzato da una notizia di rilievo e un gruppo di titoli sulla destra;

**twin-top:** impiegato quando le notizie da porre in rilievo erano due, la prima enfatizzata sulla sinistra mentre la seconda all'interno di un box più piccolo a destra;

**skirmish:** utilizzato in occasione di importanti eventi sportivi in modo da rendere l'idea di una copertina di una rivista tramite una immagine molto larga;

**war:** layout riservato ad eventi di estrema importanza, come la vincita di un campionato nazionale, dove l'intera pagina viene occupata da una immagine significativa.

Avendo abbandonato il design a tabelle, la scelta di pubblicare uno dei vari layout corrispondeva a nascondere e mostrare determinati `div` al *DOM*<sup>18</sup>.

#### 4.4 Vantaggi

Alla domanda *“quali furono i vantaggi del redesign”*, Mike Davidson rispose:

*Too many to list, really*

Fortunatamente non si limitò a questa considerazione ed elencò i principali:

**Riduzione del codice:** grazie al redesign fu possibile ridurre di circa il 50% la dimensione della home page. Ciò portò ad un risparmio di banda stimato in 2 TeraByte/giorno, 61 TeraByte/mese e 730 TeraByte/anno.

**Visualizzazione dei contenuti modulare:** separando i contenuti dalla presentazione divenne possibile visualizzare un unico blocco di contenuti (realizzato tramite il *CMS*<sup>19</sup> interno) in modo differente in base alla pagina in cui si trovava.

**Maggior controllo sulla presentazione da parte dell'utente:** con l'utilizzo della tecnologia CSS, gli utenti avrebbero avuto la possibilità di impostare il font, la dimensione dei caratteri e personalizzare le pagine stesse. Questo insieme di preferenze sarebbe poi stato salvato in un cookie inviato all'utente per mantenere le impostazioni durante la navigazione delle pagine.

<sup>18</sup>al fine di modificare dinamicamente una pagina Web tramite JavaScript è necessario che il browser utilizzi il Document Object Model (DOM), un modello di rappresentazione per documenti HTML e XML

<sup>19</sup>Content Management System

**Era la cosa giusta da fare:** l'HTML presentazionale era una tecnologia vecchia che stava venendo soppiantata dal markup strutturale unito ai CSS. Continuare con il "vecchio sistema" avrebbe aumentato la frustrazione fra gli sviluppatori del settore Web, mentre il passaggio ai fogli di stile avrebbe potuto aiutarli ad affinare le loro capacità oltre che motivarli maggiormente.

## 4.5 Limiti

I problemi che comportò il redesign del portale furono dovuti principalmente alla scarsa interoperabilità fra i browser, mentre alcune limitazioni furono dovute alla tecnologia adottata:

**Posizionamento del footer:** non fu possibile racchiudere tre colonne posizionate in modo assoluto all'interno di un div e visualizzare il footer sotto quest'ultimo. Venne quindi adottato come *workaround* quello di posizionare anche il footer in modo assoluto a una certa distanza verticale dall'inizio della pagina.

**Allineamento verticale:** Mike Davidson descrive il problema dell'allineamento verticale come "*probably the single biggest oversight by the W3C*" poiché non vi era un modo per centrare verticalmente gli elementi all'interno di un div, cosa invece molto frequente (e di semplice realizzazione) in un layout tabellare.

**Validazione:** sebbene l'aderenza agli standard fosse presa in seria considerazione da Davidson, la validazione di un sito complesso come ESPN.com che generava contenuti dinamicamente, usava meccanismi di tracking di terze parti, visualizzava contenuti tramite Flash e mostrava banner pubblicitari, avrebbe richiesto enormi sacrifici dal punto di vista delle funzionalità.

## 5 Conclusioni

I CSS hanno introdotto alcune caratteristiche uniche ed innovative, tra cui la cascata, gli pseudo-elementi e le pseudo-classi, il parsing compatibile nel futuro e i tipi di media. Fino ad ora, i CSS si sono affermati come una delle specifiche fondamentali sul Web e il numero di siti che li utilizzano è in continua crescita. In questo senso, lo sforzo per creare un linguaggio di stile per il Web ha avuto successo. Inoltre i CSS hanno parzialmente soddisfatto la loro ambizione di mantenere l'HTML un linguaggio di marcatura strutturato e di assicurare che gli utenti possano dare uno stile ai documenti.

## Riferimenti bibliografici

- [1] Andreessen, M.; NCSA Mosaic for X 0.10 available; messaggio inviato su comp.infosystems.gopher, comp.infosystems.wais, comp.infosystems, alt.hypertext e comp.windows.x 15 marzo 1993; disponibile su <http://groups.google.com/groups?selm=MARCA.93Mar14225600%40wintermute.ncsa.uiuc.edu&output=plain>
- [2] Andreessen, M.; messaggio inviato su www-talk, 17 febbraio 1994; disponibile su <http://www.webhistory.org/www.lists/www-talk.1994q1/0648.html>
- [3] Andreessen, M.; Mosaic Netscape is out the door...; messaggio inviato su www-talk 13 ottobre 1994; disponibile su <http://www.webhistory.org/www.lists/www-talk.1994q4/0187.html>
- [4] Behlendorf, B.; Re: Cascading HTML style sheets – a proposal; messaggio inviato su www-talk il 13 ottobre 1994; disponibile su <http://www.w3.org/Style/History/www.eit.com/www.lists/>
- [5] Berners-Lee, T.; Weaving the Web; Harper, San Francisco, 1999
- [6] Bos, B.; Re: Cascading HTML style sheets – a proposal; messaggio inviato su www-talk l'11 ottobre 1994; disponibile su <http://www.webhistory.org/www.lists/www-talk.1994q4/0158.html>
- [7] Cascading Style Sheets, level 1; Lie, H. W. and Bos, B.; W3C Recommendation, 17 Dicembre 1996; disponibili su <http://www.w3.org/TR/REC-CSS1-961217>
- [8] CSS1, Changes from the 17 December 1996 version: <http://www.w3.org/TR/REC-CSS1/#appendix-f>
- [9] Cascading Style Sheets, level 2; Bos, B., Lie, H. W., Lilley, C. and Jacobs, I. (editors); W3C Recommendation, 12 Maggio 1998; disponibili su <http://www.w3.org/TR/1998/REC-CSS2-19980512>
- [10] Errata in REC-CSS2-19980512: <http://www.w3.org/Style/css2-updates/REC-CSS2-19980512-errata.html>
- [11] Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification; disponibile su <http://www.w3.org/TR/CSS21/>
- [12] CSS2.1 W3C Candidate Recommendation, Processing Model; disponibile su <http://www.w3.org/TR/CSS21/intro.html#processing-model>
- [13] An Interview With Mike Davidson of ESPN (Part 1): [http://devedge-temp.mozilla.org/viewsource/2003/espn-interview/01/index\\_en.html](http://devedge-temp.mozilla.org/viewsource/2003/espn-interview/01/index_en.html)
- [14] An Interview With Mike Davidson of ESPN (Part 2): [http://devedge-temp.mozilla.org/viewsource/2003/espn-interview/02/index\\_en.html](http://devedge-temp.mozilla.org/viewsource/2003/espn-interview/02/index_en.html)
- [15] Berners-Lee, T. and Connolly, D.; HTML 2.0; RFC1866, Novembre 1995
- [16] HTML 4.0 Specification; Raggett, D., Le Hors, A. and Jacobs, I.; W3C Recommendation, 18 Dicembre 1997; disponibili su <http://www.w3.org/TR/REC-html40-971218/>

- [17] Lie, H. W.; Cascading HTML Style Sheets; proposta pubblicata il 10 Ottobre 1994; disponibile su: <http://www.w3.org/People/howcome/p/cascade.html>
- [18] Lie, H. W.; Cascading Style Sheets, PhD Thesis; disponibile su <http://people.opera.com/howcome/2006/phd/>
- [19] Eric Meyer; CSS: The Definitive Guide, 3rd Edition; O'Reilly, Novembre 2006
- [20] <http://www.webstandards.org/learn/external/css/>
- [21] Wei, P. Y.; Re: Cascading HTML style sheets - a proposal; messaggio inviato su www-talk il 24 Ottobre 1994; disponibile su <http://www.w3.org/Style/History/www.eit.com/www.lists/www-talk.1994q4/0387.html>
- [22] Lie, H. W.; CSS1 status; messaggio inviato su [www-style@w3.org](mailto:www-style@w3.org) il 25 Gennaio 1996; disponibile su <http://lists.w3.org/Archives/Public/www-style/1996Jan/0039.html>
- [23] CSS contents and browser compatibility, <http://www.quirksmode.org/css/contents.html>
- [24] <http://en.wikipedia.org/wiki/CSS3#Limitations>
- [25] Associating Style Sheets with XML documents; disponibile su <http://www.w3.org/1999/06/REC-xml-stylesheet-19990629/>
- [26] Zeldman, J.; Designing with Web Standards; New Riders, Maggio 2003