Cascading Style Sheets

Fabio Vitali

HTML e stili (1)

HTML aveva inizialmente una esplicita scala di valori:

- Contenuto
- Struttura
- Linking
- Semantica
- Presentazione

La parte presentazionale, dunque, era l'ultima in ordine di importanza della scala di valori.

Per quel che riguarda la presentazione, il prototipo WWW di Berners-Lee aveva un linguaggio di style che permetteva agli autori di definire personalmente come presentare i documenti HTML.

Analogamente, le prime versioni dei browser WWW permettevano agli **utenti** di definire queste caratteristiche.

HTML e stili (2)

Viceversa, il prototipo di Mosaic aveva pochissime opzioni per l'utente (dimensione e nome del font da usare per i testi). Addirittura, Netscape 1.0 introdusse alcuni tag (font, center) che permettevano all'**autore** di specificare caratteristiche di presentazione.

Il successo di HTML e del WWW introduce nel mondo degli autori di pagine WWW grafici e tipografi per cui è **fondamentale** gestire centralmente l'aspetto finale delle pagine. Questo significa che le indicazioni di aspetto debbono necessariamente risiedere dentro al documento, ovvero (per l'epoca) come tag e attributi HTML.

Tra la versione 2.0 e la 3.2 si assiste alla invenzione di decine di nuovi tag e attributi HTML, molti dei quali finiti poi nello standard, per la specifica di caratteristiche tipografiche e di presentazione.

CSS: Cascading Style Sheet (1)

Bert Bos (belga) e Håkon Lie (danese) sono tra i tanti propositori di un linguaggio di stylesheet per pagine HTML.

La differenza, nella loro proposta, è la parola "Cascading". Questo significa che è prevista ed incoraggiata la presenza di fogli di stile multipli, che agiscono uno dopo l'altro, in ***cascata***, per indicare le caratteristiche tipografiche e di layout di un documento HTML.

Questo permette dunque di avere controllo sia da parte dell'autore che del lettore di un documento HTML.

L'altra caratteristica di CSS che lo vide vincente fu il rendersi indipendente dalla specifica collezione di elementi ed attributi HTML, così da rendere possibile e facile il supporto di nuove versioni di HTML e anche di XML.

CSS: Cascading Style Sheet (2)

CSS level 1 (W3C Rec. dic. 1996) è un ***linguaggio di formattazione visiva***. Permette di specificare caratteristiche tipografiche e di presentazione per gli elementi di un documento HTML destinato ad essere visualizzato.

CSS level 2 (W3C Rec. Mag. 1998), invece, introduce il supporto per media multipli (es. aurali), e un linguaggio di layout sofisticato e complesso.

Il supporto dei vari browser a CSS e complesso e difficile. Infatti, tutti hanno supportato aspetti diverse ed incompatibili delle caratteristiche di CSS.

In particolare, ancora oggi nessun browser supporta per intero level 2, e d'altra parte nessuno ha mai supportato ***soltanto*** level 1 (i primi browser che supportavano CSS già avevano meccanismi per il posizionamento assoluto, che fa parte di level 2).

Usare CSS con HTML (0)

HTML prevede l'uso di stili CSS in quattro modi diversi:

- Posizionato presso il tag di riferimento
- Posizionato nel tag style
- Importato dal tag style
- Indicato dal tag link

Inoltre HTML permette di assegnare gli stili agli elementi in tre modi:

- Assegnati a tutti gli elementi di un certo tipo (il nome dell'elemento)
- Assegnati a tutti gli elementi di una certa categoria (il valore dell'attributo CLASS)
- Assegnati ad uno specifico elemento (identificato dal valore dell'attributo ID)

Usare CSS con HTML (1)

1 - Posizionato presso il tag di riferimento

```
<HTML>
  <HEAD>
  <TITLE>Bach's home page</TITLE>
  </HEAD>
  <BODY>
    <H1 STYLE="color:blue;">Home page di J.S. Bach</H1>
    <P>Johann Sebastian Bach è stato un
       compositore prolifico.</p>
  </BODY>
</HTML>
```

Usare CSS con HTML (2)

2 - Posizionato nel tag style

```
<HTML>
  <HEAD>
    <TITLE>Bach's home page</TITLE>
    <STYLE type="text/css">
      H1 { color: blue }
    </STYLE>
  </HEAD>
  <BODY>
    <H1>Home page di J.S. Bach</H1>
    <P>Johann Sebastian Bach è stato un
       compositore prolifico.</p>
  </BODY>
</HTML>
```

Usare CSS con HTML (3)

3 - Importato dal tag style

```
<HTML>
  <HEAD>
    <TITLE>Bach's home page</TITLE>
    <STYLE type="text/css">
      @import url(/style/extfile.css)
    </STYLE>
  </HEAD>
  <BODY>
    <H1>Home page di J.S. Bach</H1>
    <P>Johann Sebastian Bach è stato un
       compositore prolifico.</p>
  </BODY>
</HTML>
```

extfile.css

```
H1 {color:blue};
```

Usare CSS con HTML (4)

4 - Indicato dal tag link

```
<HTML>
  <HEAD>
    <TITLE>Bach's home page</TITLE>
    <LINK type ="text/css"
          rel  ="stylesheet"
          href ="/style/extfile.css">
  </HEAD>
  <BODY>
    <H1>Home page di J.S. Bach</H1>
    <P>Johann Sebastian Bach è stato
       compositore prolifico.</p>
  </BODY>
</HTML>
```

extfile.css

```
H1 {color:blue};
```

Usare CSS con HTML (5)

5 - Gli attributi ID e CLASS (e NAME)

- HTML 4.0 introduce due nuovi attributi per TUTTI gli elementi del documento HTML: ID e CLASS.
- ID assume un valore arbitrario purché univoco su tutto il documento. Questo permette di identificare quello specifico elemento tra tutti gli altri.
- NAME veniva usato per lo scopo che ha adesso ID, ma non era disponibile per tutti gli elementi, e soprattutto non c'era richiesta di univocità.
- CLASS assume un valore stringa qualunque. Più elementi possono condividere lo stesso valore. Questo permette di assegnare gli elementi ad una categoria, e quindi a distinguere tra paragrafi con giustificazione semantica diversa:
   …
   …
- CSS permette di assegnare regole di presentazione agli elementi per nome, per classe e per ID.

La sintassi di CSS (1)

Proprietà

- una caratteristica di stile assegnabile ad un elemento. CSS1 prevede 53 proprietà diverse, CSS2 ben 121.

```
color, font-family, margin, etc.
```

Statement

- indicazione di una proprietà CSS. Ha la sintassi

"proprietà: valore;"

```
color: blue;
font-family: "Times New Roman";
margin: 0 px;
```

La sintassi di CSS (2)

Selettore

- specificazione di un elemento o di una classe di lementi dell'albero HTML (o XML) al fine di associarvi caratteristiche CSS.

```
H1, P.heading, TD[valign]
```

Regola

- Un blocco di statement associati ad un elemento attraverso l'uso di un selettore. Ha la sintassi

```
selettore {statement; statement; ...}
```

```
H1 {
  color: blue;
  margin: 5 px;
}
```

La sintassi di CSS (3)

Regole @ (at-rules)

- 5 meta-indicazioni per la specifica di "ambiti" o meta-regole del foglio di stile: @import, @media, @font-face, @charset, @page.

```
@import "style2.css"
@media aural {
  voice-family: paul;
  stress: 20;
  richness: 90;
  cue-before: url("ping.au")
}
```

I selettori (1)

Attraverso i selettori vengono associate le regole agli elementi del documento HTML (o XML).

- **Selettore di tipo** (**E**): fa match con gli elementi E

```
BODY { font-family: Arial; font-size: 12 pt; }
H1 { font-size: 18 pt; }
P { font-size: 10 pt; }
```

- **Selettori di prossimità** (**E F E>F E + F**): fanno match con elementi F che siano discendenti, figli diretti o immediatamente seguenti ad elementi E.

```
H1+P { font-size: 11 pt; }
```

- **Selettori di attributi** (**E[foo] E[foo="bar"]**): Fanno match con gli elementi E che posseggono l'attributo specificato o in cui esso valga valore indicato.

```
A[NAME] { color: red; }
```

I selettori (2)

- **Selettori di classe** (**E.bar E#bar**): Il primo si usa solo per HTML, ed è equivalente a E[class="bar"]. Il secondo identifica gli elementi il cui attributo di tipo ID vale "bar".

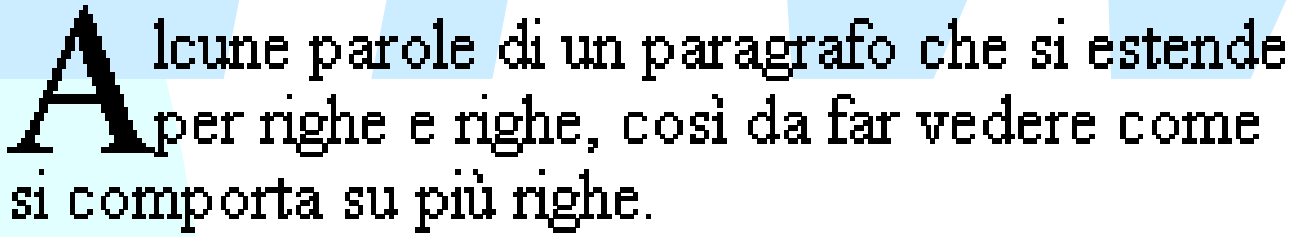
```
H1.important { font-size: 24 pt; }
P#note1 { font-size: 9 pt; }
```

- **Selettori di pseudo-classi** (**E:first-child E:link E:visited E:active E:hover E:focus E:lang(c)**): Fanno match con elementi E solo in certi casi ed in certe situazioni
  - **first-child**: vero se l'elemento è il primo figlio di E.
  - **link, visited**: vero se l'elemento E è, rispettivamente, un link non ancora visitato o un link già visitato.
  - **hover, active, focus**: vero se, rispettivamente, sull'elemento E passa sopra il mouse, il mouse è premuto, o il controllo è selezionato per accettare input.
  - **lang(c)**: vero se l'elemento ha selezionata la lingua c.

I selettori (3)

- **Selettori di pseudo-elementi** (**E:first-line E:first-letter E:before E:after**): Vengono attivati in corrispondenza di certe parti degli elementi E.
  - **before, after**: vero prima e dopo il contenuto dell'elemento E.
  - **first-line**: vero per la prima riga dell'elemento E.
  - **first-letter**: vero per la prima lettera di un elemento.

```
P:first-letter {
      font-size: 300%;
      float:left;
}
```



- *****: Selettore universale (fa match con tutto)
- **,**: raggruppamento di selettori: selettori diversi possono usare lo stesso blocco se separati da virgola.

Modello visuale di CSS (1)

La **visualizzazione** di un documento con CSS avviene identificando lo spazio di visualizzazione di ciascun elemento del documento.

Ogni elemento è definito da una ***scatola*** all'interno del quale sta il contenuto. Le scatole sono in relazione alle altre come segue:

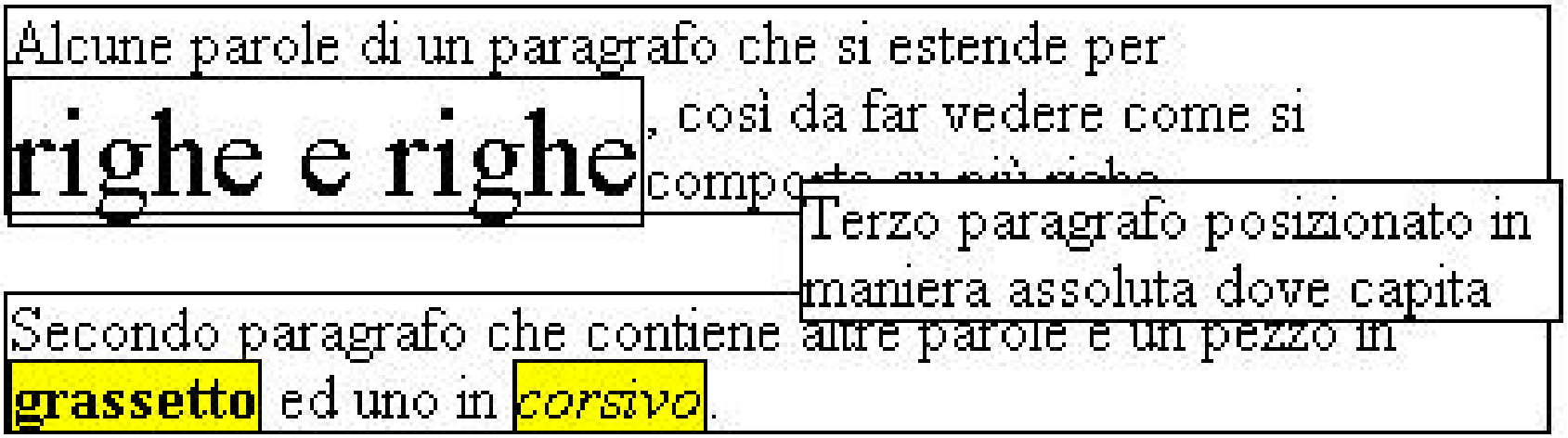
- Le scatole degli elementi **contenuti** stanno dentro alla scatole dell'elemento genitore.
- **Flusso normale di tipo blocco**: le scatole sono poste l'una sopra l'altra in successione verticale (come paragrafi).
- **Flusso normale di tipo inline:** le scatole sono poste l'una accanto all'altra in successione orizzontale (come parole della stessa riga)
- **Flusso di tipo float**: le scatole sono poste all'interno del contenitore e poi spostate all'estrema sinistra o destra della scatola, lasciando che le altre scatole vi girino intorno.
- **Posizionamento assoluto**: le scatole vengono poste nella posizione indicata indipendentemente dal flusso e da quel che la zona già visualizza (eventualmente nascondendo ciò che sta sotto).

Modello visuale di CSS (2)

Alcune proprietà controllano il tipo di posizionamento e di scatola:

- DISPLAY (inline | block | list-item | run-in | ... | none): il tipo di scatola da utilizzare per l'elemento: un blocco, un inline, una lista, una cella di tabella, ecc.
- POSITION (static | relative | absolute | fixed): il posizionamento rispetto al flusso del documento.
- FLOAT (left | right | none): un float è una scatola scivolata all'estrema destra o sinistra del contenitore muovendo le altre per farle posto.
- Z-INDEX:la posizione nello stack di scatole potenzialmente sovrapposte. Il valore più alto è più vicino al lettore, e quindi nasconde gli altri. N.B.: per default il valore di background delle scatole è trasparente.
- TOP, BOTTOM, LEFT, RIGHT: coordinate della scatola
- WIDTH, HEIGHT: dimensioni usabili invece di right e bottom.

Un esempio di posizionamento (1)

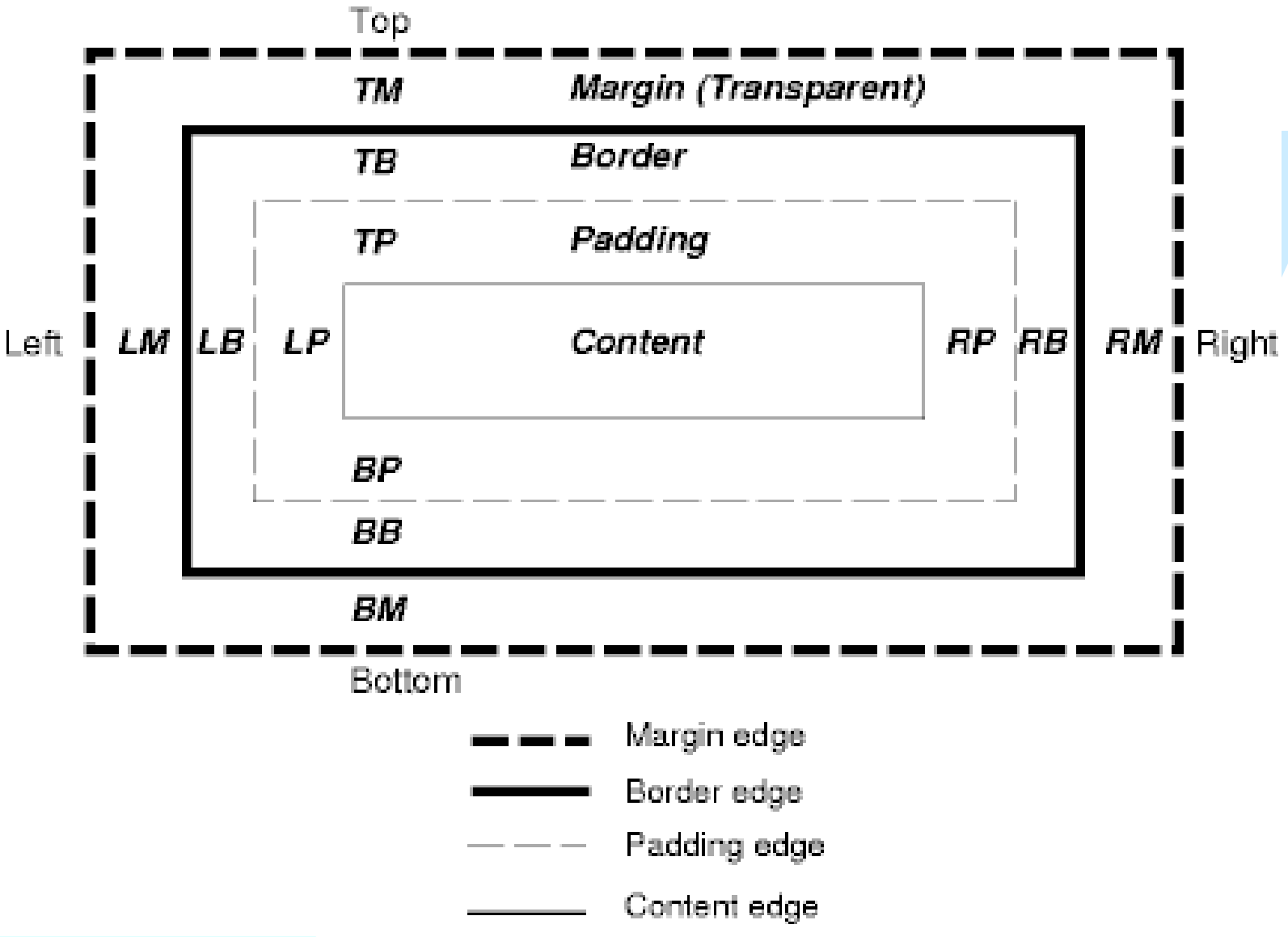


Un esempio di posizionamento (2)

```
p.abs { position: absolute; top: 40px; left: 210px;
        width: 190px; background:white;
        border-style: solid; border-width: 1px;}
p { display: block; border-style: solid;
    border-width: 1px; }
b,i { display:inline; border-style: solid;
      border-width: 1px; background:yellow;}
span.left { border-style: solid; border-width: 1px;
            float:left; font-size: 200%;}
```

```
<P>Alcune parole di un paragrafo che si estende per
<span class="left">righe e righe</span>, cos&igrave; da
far vedere come si comporta su pi&ugrave; righe.</P>
<P>Secondo paragrafo che contiene altre parole e un
pezzo in <b>grassetto</b> ed uno in <i>corsivo</i>.</p>
<p class="abs">Terzo paragrafo posizionato in maniera
assoluta dove capita </p>
```

Elementi della scatola



Elementi della scatola (2)

Margini: la regione che separa una scatola dall'altra, necessariamente trasparente.

- margin-top, margin-bottom, margin-left, margin-right: dimensioni del margine della scatola.

Border: la regione ove visualizzare un bordo per la scatola.

- border-top, border-bottom, border-left, border-right, border-width, border-color: dimensioni ed aspetto del bordo.
- border-style: può assumere come valori **none, dotted, dashed, solid, double, groove, ridge, inset, outset.**

Padding: la regione di respiro tra il bordo della scatola ed il contenuto. Ha il colore dello sfondo.

- padding-top, padding-bottom, padding-left, padding-right: dimensioni del padding della scatola.

Content: la regione dove sta il contenuto dell'elemento.

- background-color, background-image, background-repeat, background-attachment, background-position: colore, immagine e meccanismo di ripetizione dell'immagine di sfondo della scatola.

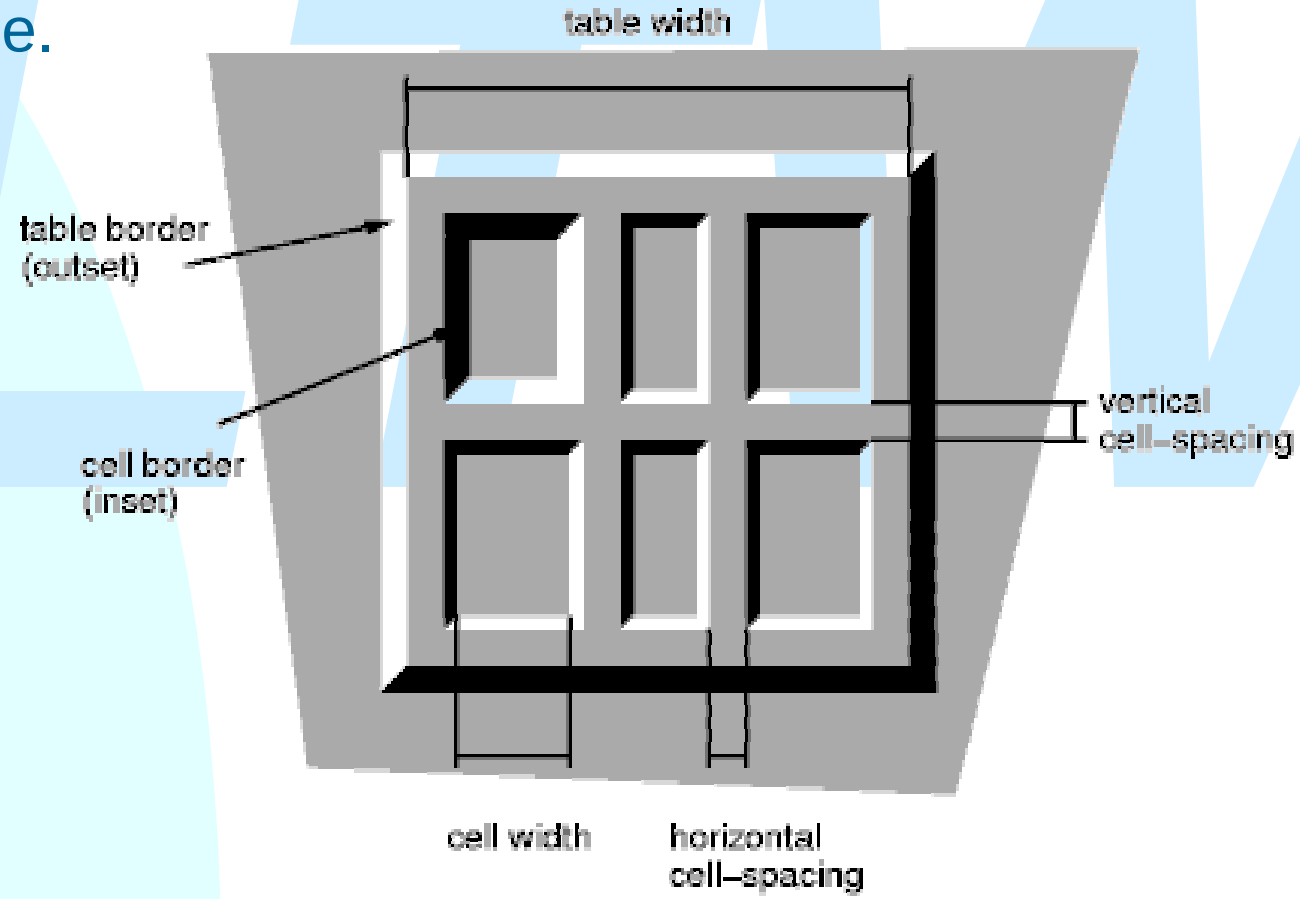
Il testo

Del testo è possibile controllare sia gli aspetti relativi al font che quelli relativi all'organizzazione del testo nella scatola di riferimento:

- font-family: il/i nomi del/dei font
- font-style (normal | italic | oblique), font-variant (normal | small-caps), font-weight (normal | bold | bolder | lighter | 100<-> 900), font-stretch (normal | wider | narrower | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded): caratteristiche del font
- text-indent, text-align, line-height: indentazione, allineamento e interlinea delle righe della scatola.
- text-decoration (none | underline | overline | line-through | blink), text-shadow: ulteriori stili applicabili al testo
- letter-spacing e word-spacing: spaziatura tra lettere e tra parole
- text-transform (capitalize | uppercase | lowercase | none): trasformazione della forma delle lettere.
- white-space (normal | pre | nowrap): specifica la gestione dei ritorni a capo e del collassamento dei whitespace all'interno di un elemento.

Tabelle

CSS permette di definire proprietà sofisticate per gli elementi di una tabella, in termini di scatole per gruppi di colonne, colonne, gruppi di righe, righe, e singole celle.



Font: descrizione astratta

Elaborazioni tipografiche sofisticate possono richiedere l'uso di font che forse sono e forse NON sono disponibili sul computer dell'utente.

CSS permette di specificare font anche non disponibili, fornendo un meccanismo per scaricarlo dal server…

```
@font-face {
  font-family: "Robson Celtic";
  src: url("http://site/fonts/rob-celt")
       format("truetype");
}
```

… oppure per trovare tra i font locali quello più simile a quello specificato:

```
@font-face {
   panose-1: 2 4 5 2 5 4 5 9 3 3;
   font-family: Alabama, serif;
   font-style:  italic, oblique;
}
```

Proprietà aurali

CSS2 fornisce un grande numero di caratteristiche aurali specificabili per gli elementi di un documento HTML, pronte per essere pronunciate da un sintetizzatore vocale con l'aiuto di "icone uditive" (ad es. clip predefinite).

E' ovvio che convertire il documento in testo e leggere semplicemente le parole non ottiene gli effetti desiderati. E' allora possibile utilizzare le proprietà aurali di CSS, che identificano direzione dei suoni, organizzazione temporale dei suoni, e variazioni nel parlato sintetizzato (voce, frequenza, velocità inflessione, ecc.).

- Volume properties: 'volume'
- Speaking properties: 'speak' (normal | none | spell-out)
- Pause properties: 'pause-before', 'pause-after', and 'pause'
- Cue properties: 'cue-before', 'cue-after', and 'cue'
- Mixing properties: 'play-during'
- Spatial properties: 'azimuth' and 'elevation'
- Voice characteristic properties: 'speech-rate', 'voice-family', 'pitch', 'pitch-range', 'stress', and 'richness'
- Speech properties: 'speak-punctuation' and 'speak-numeral', 'speak-header'

Altri tipi di media

In CSS è possibile specificare regole di stile specifiche per il tipo di media utilizzato.

Attualmente CSS permette di specificare i seguenti tipi di media: **aural** (sintetizzatore di voce), **braille** (terminale braille elettronico), **embossed** (pagina in braille a rilievo), **handheld** (agenda elettronica dallo schermo piccolo), **print** (carta stampata), **projection** (proiettore da presentazioni), **screen** (schermo di computer grafico e a colori), **tty** (terminale a carattere), **tv** (schermo televisivo, di dimensioni anche grandi ma con risoluzione pessima).

Attraverso @media è possibile specificare regole diverse per ogni media:

```
@media print {
  BODY { font-size: 10pt }
}
@media screen {
  BODY { font-size: 12pt }
}
@media screen, print {
  BODY { line-height: 1.2 }
}
```

Tipi di dati

I valori delle proprietà in CSS possono assumere valori di una grande quantità di tipi. I più importanti:

- **Interi e reali**: rappresentano numeri assoluti (es. volume o z-index)
- **Lunghezze**: rappresentano misure numeriche espresse in una determinata unità di misura. Tra le unità di misura: **em, px, pt, in, cm, mm**.
- **Percentuali**: rappresentano una misura relativa rispetto all'ambiente circostante (ad esempio la scatola contenitore).
- **URI**: **`url(http://…)`**
- **Colori**: o per nome (gli stessi di HTML), oppure per codice RGB, secondo la sintassi **`rgb(XX,XX,XX)`**, dove XX è un numero tra 0 e 255.
  - Bianco: **`white`** oppure **`rgb(255,255,255)`**
- **Stringhe**: una stringa posta tra virgolette semplici o doppie. Si usa la barra rovesciata per includere le virgolette nella stringa.
  - "Io mi chiamo \"Fabio\""

Valori ereditati

Se non viene specificata una proprietà, CSS assume un valore di default.

A parte pochi casi, questo è sempre "inherit". Questo significa che la proprietà assume lo stesso valore che ha nella scatola contenitore dell'elemento in questione. Ad esempio l'elemento em qui avrà il colore rosso.

```
<p style="color:red;">
  Qui &egrave; <em>in corsivo</em> e qui no.
</p>
```

Tra i valori non ereditati:

- **display** (per HTML è sempre il valore naturale dell'elemento, block per P o H1, inline per B, I o A, mentre per XML è inline)
- **background** (sempre transparent)

La cascata

Come si è detto, CSS ha avuto successo perché permette sia agli autori che agli utenti di esprimere preferenze sulle regole di presentazione.

E' possibile cioè definire regole multiple per gli stessi elementi, e adottare un meccanismo a cascata per la loro applicazione:

- **User Agent**: il browser definisce (o esplicitamente o implicitamente codificandole nel software) le regole di default per gli elementi dei documenti.
- **User**: l'utente può fornire un ulteriore foglio di stile per indicare regole di proprio piacimento. Tipicamente è una funzione del browser
- **Author**: l'autore delle pagine fornisce, nei modi visti in precedenza, i fogli di stile del documento specifico.
- **Regole `!important`** : Quando una regola utente (tipicamente) è seguita dalla keyword **`!important`**, essa sopravanza una analoga regola di senso diverso dell'autore.

```
P { font-size: 18pt !important }
```

Forme abbreviate

In molti casi è possibile riassumere in un'unica proprietà i valori di molte proprietà logicamente connesse.

Si usa una sequenza separata da spazi di valori, secondo un ordine prestabilito. Se si mette un valore solo esso viene assunto da tutte le proprietà individuali. Ad esempio:

- margin per margin-top, margin-left, margin-bottom, margin-right
- border per border-top, border-left, border-bottom, border-right
- padding per padding-top, padding-left, padding-bottom, padding-right
- font per font-style, font-variant, font-weight, font-size, line-height, font-family

```
P { font: bold italic large Palatino, serif }
BODY { margin: 1em 2em 3em 4em; }
```

```
BODY {
  margin-top: 1em;
  margin-right: 2em;
  margin-bottom: 3em;
  margin-left: 4em;
}
```

```
BODY { padding: 2em; }
```

```
BODY {
 padding-top: 2em;
 padding-right: 2em;
 padding-bottom: 2em;
 padding-left: 2em;
}
```

Conclusioni

CSS vuole risolvere la separazione tra contenuto e presentazione in HTML (e in XML)

Fornisce un linguaggio completo per layout di documenti e tipografia sofisticata. Tuttavia manca di un meccanismo di riordinamento e riuso del contenuto (che invece ha XSL).

Usa una sintassi tutta sua, priva di riferimenti con altri linguaggi, usabile sia all'interno del documento che in un documento autonomo.

Le implementazioni di CSS sono quanto di più variabile si possa trovare. Non esiste un browser che implementi tutto CSS esattamente, e ci sono variazioni tra S.O. e S.O., versione e versione, browser e browser.

In definitiva, è un livello ulteriore di complessità nella progettazione delle pagine, a cui corrisponde una notevole quantità di lavoro per ottenere risultati prevedibili su tutti i browser.

Riferimenti

- B. Bos, H. Lie, C. Lilley, I. Jacobs, ***Cascading Style Sheets, level 2***, W3C Recommendation 12 May 1998, http://www.w3.org/TR/REC-CSS2
- H. Lie, B. Bos, ***Cascading Style Sheets, level 1***, W3C Recommendation 17 Dec 1996, revised 11 Jan 1999, http://www.w3.org/TR/REC-CSS1
- T. Markula, ***Introduction to CSS***, http://www.nic.fi/~tapio1/Teaching/index2.php3