

Laboratorio di Informatica II

Riccardo Ortale



CSS

Lezioni 5 - 6

Introduzione

- ◆ Oggi vediamo CSS (Cascading Style Sheet)
 - L'esigenza di uno stile per HTML
 - L'introduzione di CSS
 - Come usare CSS in HTML
 - La sintassi degli statement CSS
 - Il modello visuale di CSS
 - Il modello aurale di CSS

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 comprendeva un linguaggio di stile, che permetteva ai lettori 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 i quali 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, in gran parte confluiti 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 fruitore 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 è complesso e difficile. Infatti, tutti hanno supportato aspetti diverse ed incompatibili delle caratteristiche di CSS.
- ◆ In particolare, ancora oggi i browser non supportano per intero level 2, e d'altra parte nessuno ha mai supportato *soltanto* level 1 (i primi browser che supportavano CSS includevano anche meccanismi per il posizionamento assoluto, che sono caratteristici 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 CSS 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)

Il concetto di cascata

- ◆ Gli attributi di un elemento vengono presi non da uno (il primo, l'ultimo, ecc.) dei fogli di stile, ma composti dinamicamente sulla base del contributo di tutti, in cascata.
- ◆ Ad esempio, avendo tre fogli di stile, che riportano ciascuno una delle seguenti regole,
 - `P { font-family: Arial; font-size: 12 pt; }`
 - `P {color: red; font-size: 11 pt; }`
 - `P {margin-left: 15 pt; color: green;}`
- ◆ ... gli attributi dell'elemento P saranno equivalenti a:
 - `P {
font-family: Arial;
font-size: 11 pt;
margin-left: 15 pt;
color: green;
}`

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 un`
- ◆ `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 uno 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 certa categoria e, quindi, di distinguere tra elementi uguali, ma appartenenti a categorie diverse.
 - Nell'esempio si distingue tra paragrafi con giustificazione semantica diversa:
`<p class="spiegazione"> ... </p>`
`<p class="esempio"> ... </p>`
 - 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 elementi, dell'albero HTML (o XML), a cui applicare statement 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 ad esso sia assegnato un 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 in cui l'attributo di tipo ID assume valore "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)

◆ Effetto ottenuto senza usare immagini, ma semplicemente P:first-letter

- **Selettori di pseudo-elementi (E:first-line E:first-letter E:before E:after):** Si attivano soltanto 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;  
}
```

 Alcune parole di un paragrafo che si estende per righe e righe, così da far vedere come si comporta su più righe.

- *: Selettore universale (fa match con tutto)

- ,: raggruppamento di selettori: selettori diversi possono usare lo stesso blocco se separati da virgola.

Esempi d'uso dei selettori

- ◆ **p em { /*...*/ }**
 - Questa regola è applicata a tutte le porzioni di testo enfattizzato racchiuse in un paragrafo.
- ◆ **div#header h2 { /*...*/ }**
 - Si applica agli elementi `<h2>` inclusi nella sezione della pagina individuata dall' id *header*.
- ◆ **div.news p { /*...*/ }**
 - Tutti i paragrafi inseriti nei div con classe *news* sono presentati attraverso questa regola.
- ◆ **div#navigation ul#mainindex a { /*...*/ }**
 - In questo caso la regola è rivolta a tutti i link della lista individuata dall' id *mainindex* e contenuta nella sezione *navigation* della pagina.

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*, al cui interno risiede il contenuto. Le scatole sono in relazione alle altre come segue:
 - Le scatole degli elementi **contenuti** risiedono nella scatola dell'elemento genitore.
 - **Flusso normale di tipo blocco (o verticale)**: le scatole sono poste l'una sopra l'altra in successione verticale (come paragrafi e tabelle). Ogni blocco è collocato su una nuova riga.
 - **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)

- ◆ E' possibile inserire un nuovo blocco con flusso verticale mediante i tag *div*, *p*, *h*
- ◆ E' possibile inserire un nuovo inline mediante i tag *span*, *b*, *i*.

Modello visuale di CSS (3)

- ◆ 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, con spostamento delle altre.
 - **Z-INDEX**: la posizione nello stack di scatole potenzialmente sovrapposte. Il valore più alto è più vicino al lettore, e quindi nasconde le altre. 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`.

La proprietà Display

- ◆ **La lista dei possibili valori è lunghissima. Solo alcuni di essi sono però di normale utilizzo e supportati:**
 - **inline.** Valore di default. L'elemento assume le caratteristiche degli elementi inline.
 - **block.** L'elemento viene trattato come un elemento blocco.
 - **list-item.** L'elemento si trasforma in un elemento lista.
 - **run-in.** L'elemento viene incorporato e inserito all'inizio del blocco seguente. L'effetto dovrebbe essere simile a quello visto per le liste quando si usa il valore `inside` per il marcatore. Il valore è supportato solo da Opera 5/6 e parzialmente da Explorer 5 Mac.
 - **compact.** L'elemento viene piazzato al fianco di un altro. Non supportato da nessun browser.
 - **marker.** Questo valore fa sì che il contenuto generato con gli pseudo-elementi `:before` e `:after` sia trattato come un marcatore di liste. Non supportato da nessun browser.
 - **none.** L'elemento non viene mostrato. Più precisamente, è come se non fosse nemmeno presente nel documento, in quanto non genera alcun box. Diverso l'effetto della proprietà `visibility:hidden`, che invece si limita a nascondere l'elemento.

La proprietà Position (1)

- ◆ **Position** è la proprietà fondamentale per la gestione della posizione degli elementi, di cui determina la modalità di presentazione sulla pagina. Si applica a tutti gli elementi e non è ereditata.
- ◆ **Valori**
 - **Static.** E' il valore di default, quello predefinito per tutti gli elementi non posizionati secondo un altro metodo. Rappresenta la posizione normale che ciascuno di essi occupa nel flusso del documento.
 - **Absolute.** L'elemento, o meglio, il box dell'elemento viene rimosso dal flusso del documento ed è posizionato in base alle coordinate fornite con le proprietà **top**, **left**, **right** o **bottom**. Il posizionamento avviene sempre rispetto al box contenitore dell'elemento. Questo è rappresentato dal primo elemento antenato (ancestor) che abbia un posizionamento diverso da static. Se tale elemento non esiste il posizionamento avviene in base all'elemento radice HTML, che in condizioni standard coincide con l'area del browser che contiene il documento e che ha inizio dall'angolo superiore sinistro di tale area. Un elemento posizionato in modo assoluto scorre insieme al resto del documento

La proprietà Position (2)

- **Fixed.** Usando questo valore il box dell'elemento viene, come per absolute, sottratto al normale flusso del documento. La differenza sta nel fatto che per fixed il box contenitore è sempre il cosiddetto viewport. Con questo termine si intende la finestra principale del browser, ovvero l'area del contenuto. Altra differenza fondamentale: un box posizionato con fixed non scorre con il resto del documento. Rimane, appunto, fisso al suo posto. L'effetto è lo stesso che si può ottenere con l'uso dei frame, in cui una parte della pagina rimane fissa e il resto scorre. Solo che qui il documento è solo uno. Il valore non è supportato da Explorer su Windows
- **Relative.** L'elemento viene posizionato relativamente al suo box contenitore. In questo caso il box contenitore è il posto che l'elemento avrebbe occupato nel normale flusso del documento. La posizione viene anche qui impostata con le proprietà **top**, **left**, **bottom**, **right**. Ma qui esse non indicano un punto preciso, ma l'ammontare dello spostamento in senso orizzontale e verticale rispetto al box contenitore.

Un esempio di posizionamento (1)

Alcune parole di un paragrafo che si estende per
righe e righe, così da far vedere come si
comporta su più righe.

Terzo paragrafo posizionato in
maniera assoluta dove capita
un pezzo in

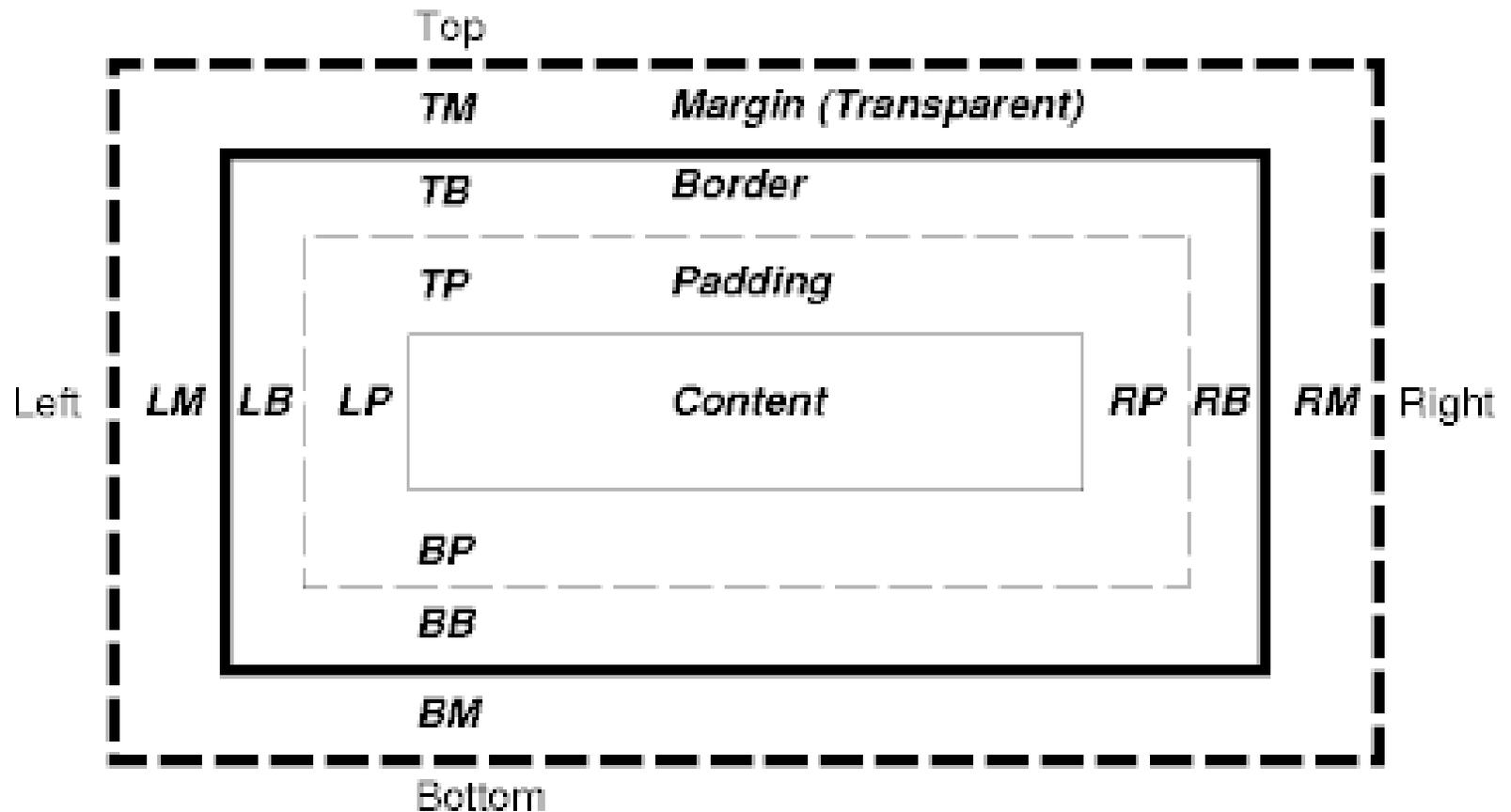
Secondo paragrafo che contiene altre parole e un pezzo in
grassetto ed uno in *corsivo*.

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, così da far vedere come`
`si comporta su più righe.</P>`
- ◆ `<P>Secondo paragrafo che contiene altre parole e un pezzo in`
`grassetto ed uno in <i>corsivo</i>.</p>`
- ◆ `<p class="abs">Terzo paragrafo posizionato in maniera assoluta`
`dove capita </p>`

Elementi della scatola



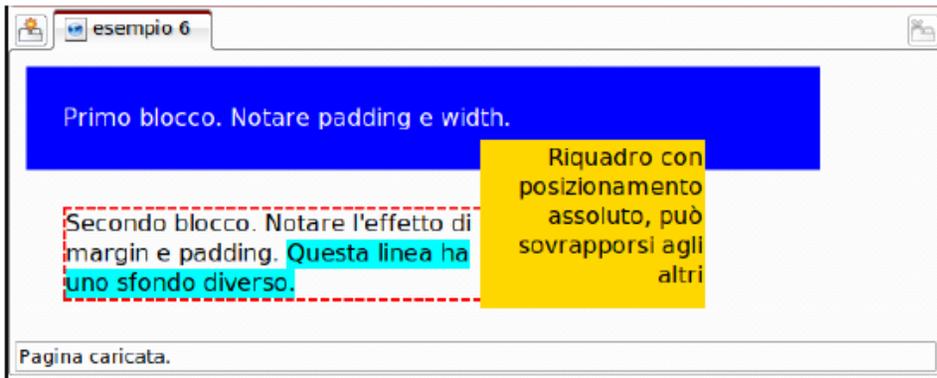
- Margin edge
- Border edge
- - - Padding edge
- Content edge

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 dove 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.

Altro esempio (1)

```
...
<link type="text/css" rel="stylesheet"
      href="./css_1.css" />
...
<div id="uno">
  Primo blocco. Notare padding e width.
</div>
<div id="due">
  Secondo blocco. Notare
  l'effetto di margin e padding.
  <span style="background-color:cyan;">
  Questa linea ha uno sfondo diverso. </span>
</div>
<div id="tre">
  Riquadro con posizionamento
  assoluto, può sovrapporsi agli altri
</div>
...
```



CSS_1.CSS

```
#uno {
  background-color: blue;
  padding: 25px;
  width: 80%;
  color: white;
}

#due {
  margin: 25px;
  border: 2px red dashed;
  padding: 0px;
  width: 300px;
}

#tre {
  background-color: gold;
  position: absolute;
  top: 60px; left: 320px;
  width: 150px; height: 110px;
  padding: 2px;
  text-align: right;
}
```

Altro esempio (2)

```
...  
<link type="text/css" rel="stylesheet"  
      href="./css_2.css" />
```

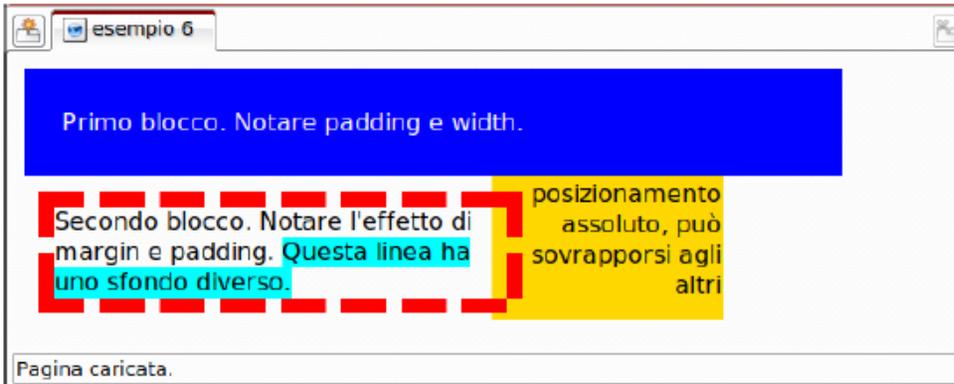
```
...  
<div id="uno">  
  Primo blocco. Notare padding e width.  
</div>
```

```
<div id="due">  
  Secondo blocco. Notare  
  l'effetto di margin e padding.  
  <span style="background-color:cyan;">  
  Questa linea ha uno sfondo diverso. </span>  
</div>
```

```
<div id="tre">  
  Riquadro con posizionamento  
  assoluto, può sovrapporsi agli altri  
</div>
```

css_2.css

```
#uno {  
  background-color: blue;  
  padding: 25px;  
  width: 80%;  
  color: white;  
}  
  
#due {  
  margin: 10px;  
  border: 10px red dashed;  
  padding: 0px;  
  width: 300px;  
}  
  
#tre {  
  z-index: -1;  
  background-color: gold;  
  position: absolute;  
  top: 60px; left: 320px;  
  width: 150px; height: 110px;  
  padding: 2px;  
  text-align: right;  
}
```



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.

Lo stile dei font (1)

- ◆ *font-family*: indica il nome del font da utilizzare per il testo e le alternative nel caso in cui il font non sia presente nella macchina ospite. E' buona norma indicare come ultima alternativa un carattere standard quale: serif (con grazie), sans-serif (senza grazie), monospaced (larghezza fissa)
 - Nota: le grazie sono quegli abbellimenti tipografici delle lettere, che rendono il testo più leggibile
 - Esempio

```
p { font-family: "tahoma", "arial sans-serif"; }
```

Quando la pagina viene caricata, il browser tenta di usare il primo font (tahoma). Se questo ultimo non è disponibile, usa il secondo (arial sans-serif).
- ◆ *font-style* (normal | italic | oblique): imposta in corsivo o in obliquo un testo
- ◆ La proprietà *font-variant* è usata per scegliere tra le varianti *normal* e *small-caps* di un font: il valore *small-caps* crea l'effetto maiuscoletto, *normal* non introduce variazioni

Lo stile dei font (2)

- ◆ *font-size*: specifica le dimensioni del font; 4 possibilità
 - grandezze assoluta in un insieme di valori predefiniti (xx-small | x-small | small | medium | large | x-large | xx-large)
 - grandezza assoluta in punti, e.g. *font-size: 12pt*;
 - grandezza relativa alle dimensioni predefinite del testo nel blocco contenitore (smaller | larger)
 - grandezza percentuale rispetto al valore predefinito per il testo nel blocco contenitore, e.g. *font-size: 110%*;
- ◆ *font-weight* (*normal* | *bold* | *bolder* | *lighter* | 100 – 900)
 - specifica il “peso” del font, cioè quanto è marcato il tratto
 - *bolder* e *lighter* sono relativi al valore predefinito
 - gli altri valori sono pesi assoluti
 - 100-900 è una scala che va dal peso minimo al massimo; non tutti i valori da 100 a 900 sono disponibili per tutti i font, il peso effettivamente adottato è il più vicino disponibile

Lo stile del testo

- ◆ *text-indent*: imposta l'indentazione in valore assoluto o percentuale
- ◆ *text-align* (*left* | *right* | *center* | *justify*): imposta l'allineamento del testo nel blocco di riferimento
- ◆ *line-height*: permette di impostare l'interlinea
- ◆ *text-decoration* (*none* | *underline* | *overline* | *line-through* | *blink*): permette di decorare il testo
 - i possibili valori sono: non decorato, sottolineato, soprallineato, cancellato, lampeggiante
- ◆ *letter-spacing* e *word-spacing*: spaziatura tra lettere e tra parole, specificata in valore
 - assoluto, e.g. *letter-spacing*: 1em
 - relativo al valore predefinito, e.g. *word-spacing*: +0.2em

Tabelle (1)

- ◆ 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.

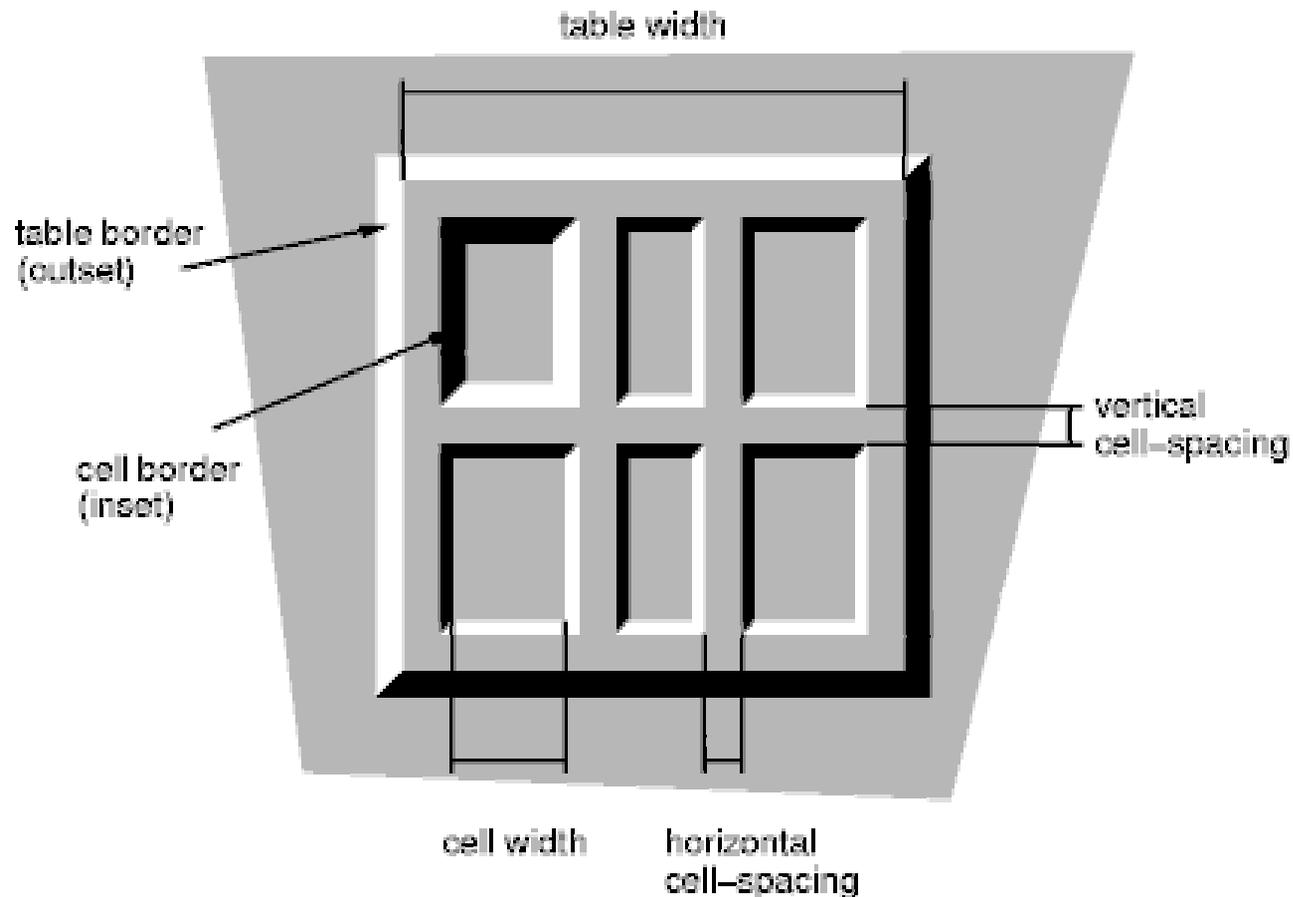


Tabelle (2)

- ◆ Gruppi di colonne, colonne, gruppi di righe, righe e singole celle hanno proprietà analoghe a quelle del box model per i blocchi

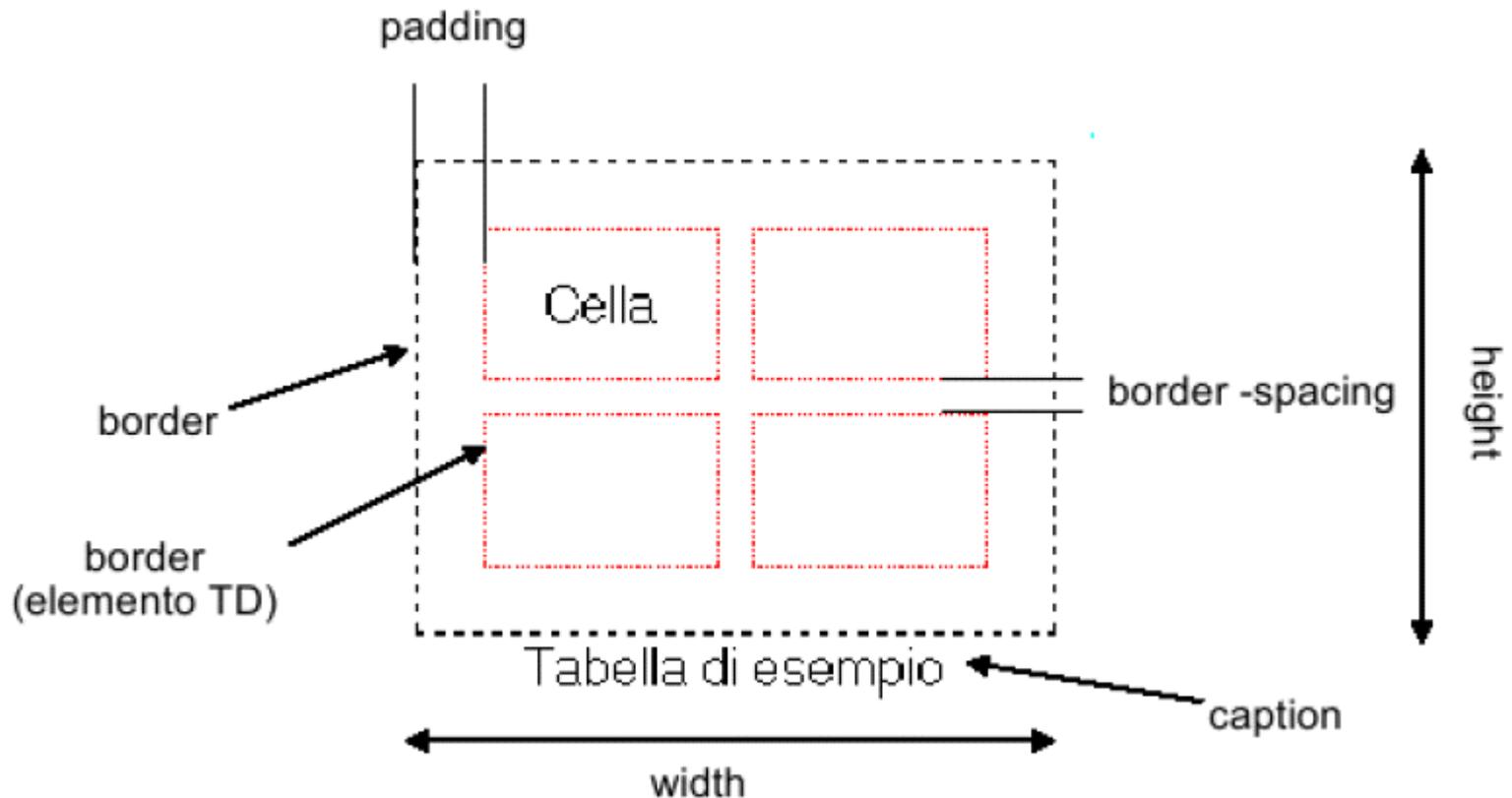


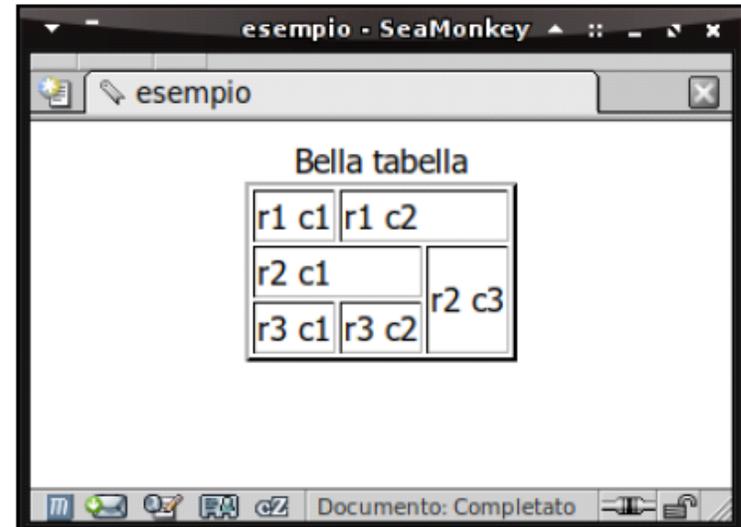
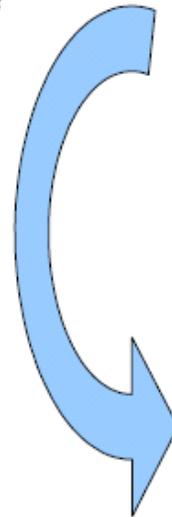
Tabelle (3)

Applichiamo il seguente foglio di stile all'esempio di tabella visto in precedenza:

```
table {  
  width: 100%;  
  background-color: #ffb400;  
  border: 10px double black;  
  font-family: "Forte" serif;  
}
```

```
table caption {  
  color: green;  
  font-size: 150%;  
  font-variant: small-caps;  
}
```

```
td {  
  color: blue;  
  background-color: #ffff80;  
  text-align: center;  
  padding: 5px;  
}
```



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 le 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 ed 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.**
 - **em** è un'unità di misura relativa, che equivale all'altezza media di un carattere per un dato font
 - **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\""

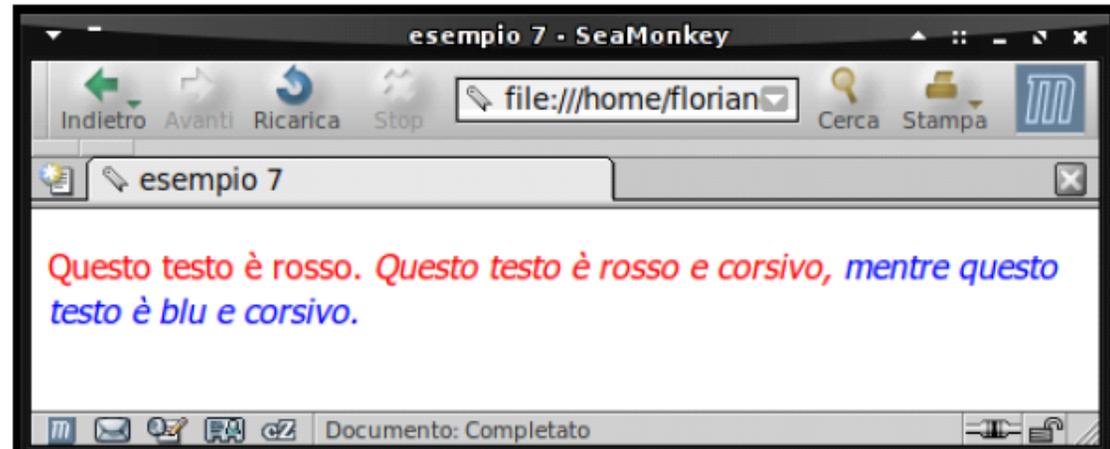
Valori ereditati (1)

- ◆ Se non viene specificata una proprietà, CSS assume un valore di default.
- ◆ A parte pochi casi, questo è sempre "inherit". Ciò significa che la proprietà assume (eredita) lo stesso valore posseduto dalla analoga proprietà della scatola contenitore in cui è inserito l'elemento in questione. Ad esempio, l'elemento *em* qui avrà il colore rosso.
- ◆ `<p style="color:red;">`
- ◆ `Qui è in corsivo 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)

Valori ereditati (2)

- ◆ Un esempio leggermente più complicato

```
<p style="color:red;">
Questo testo &egrave; rosso.
<em> Questo testo &egrave;
  rosso e corsivo,
  <span style="color: blue;">
  mentre questo testo
  &egrave; blu e corsivo.
</span>
</em>
</p>
```



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 ed 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 un'analogha 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 specifica un unico valore, questo 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, caratterizza XSL).
- ◆ Usa una sintassi propria, priva di riferimenti con altri linguaggi, applicabile sia all'interno del documento stesso, 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.

Una semplice applicazione

- ◆ Realizzare una pagina con il contenuto ed il layout seguenti

1. Area dell'intestazione	
2. Area del menu principale	
3. Menu laterale	4. Contenuto principale della pagina
Home Page Voce menu 1 Voce menu 2	Generalmente si usa suddividere il documento in cinque sezioni fondamentali: <ul style="list-style-type: none">1. Header (intestazione)2. Menu3. Menu laterale4. Corpo5. Trailer (coda) <p>La proprietà <i>float</i> rende un elemento fluttuante a destra o a sinistra del suo contenitore. Gli elementi adiacenti vengono affiancati sul suo lato libero e continuano disponendosi <i>come se questo fosse un elemento di tipo inline</i>.</p> <p>La proprietà <i>clear</i> nella sezione trailer consente di impedire ad un elemento di avere elementi float su uno o entrambi i lati. I possibili valori sono <i>right</i>, <i>left</i> o <i>both</i>. Il suo uso, spesso necessario in un layout che usa <i>float</i>, può purtroppo avere effetti indesiderati.</p> <p>Se si modificasse la proprietà <i>background</i> del blocco menubar, ci sarebbero problemi di visualizzazione quando le lunghezze del menu e del corpo del testo non coincidessero. Per questo motivo il colore di sfondo della barra laterale deve essere uguale a quello del container.</p>
5. Area di coda della pagina	

Il documento HTML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title> Creazione di un layout solo con CSS senza tabelle </title>
    <style type="text/css"> @import url(layout1.css); </style>
  </head>
  <body>
    <div class="container">
      <div class="header">      <h1>1. Area dell'intestazione</h1>      </div>
      <div class="menu">      <p>2. Area del menu principale</p>      </div>
      <div class="menubar">
        <p>3. Menu laterale</p>
        <ul>
          <li><a href="#">Home Page</a></li>
          <li><a href="#">Voce menu 1</a></li>
          <li><a href="#">Voce menu 2</a></li>
        </ul>
      </div>
      <div class="content">      <p>4. Contenuto principale [...]</p>      </div>
      <div class="trailer">      5. Area di coda della pagina      </div>
    </div>
  </body>
</html>
```

Il foglio di stile

```
body { font-family: "century schoolbook", "times new roman", serif; }
/* Stile del contenitore principale */
div.container {
  width: 90%; margin: .5em auto; /* centra in browser diversi da IE*/
  border: 1px solid black; text-align: left;
  background: yellow; /* come il blocco float per il menu laterale */
}
h1 { margin: 0px; padding: 0px; color: white; }
/* Stile dell'header del documento */
.header { padding: 1em; background: #99B7DB; border-bottom: 1px solid black; }
/* Stile del menu principale */
.menu {
  background: green; margin-bottom: 0px; padding-top: .5em;
  text-align: center; border-bottom: 1px solid black; }
/* Stile del menu principale */
.menubar { float: left; width: 20%; margin-top: 0px; background: yellow; }
.menubar ul { list-style-type: none; margin-left: 0px; padding-left: 1em; }
.menubar a:hover { color: red; }
.menubar a { font-weight: bold; text-decoration: none; /* link non sottolin.*/ }
/* Stile del corpo del testo */
.content { background: white; margin-left: 20%;
  padding: 1em; border-left: 1px solid black; }
/* Stile del footer del documento */
.trailer { clear: both; /* evita blocchi float ai lati */
  margin: 0px; padding: .5em; background: #99b7db;
  font-size: 75%; text-align: right; border-top: 1px solid black; }
```

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>