

I fogli di stile CSS



Cascading Style Sheets

La presentazione accessibile dei contenuti

Prima dei CSS

Il “primo Web” era tendenzialmente...

- **Fondato sui contenuti piuttosto che sul loro formato**
- **Graficamente “recintato” dai limiti di HTML**



Prima dei CSS

Tentativi di rendere “bello” il Web

- **Uso di estensioni proprietarie dell'HTML**
- **Conversione di testo in immagini**
- **Uso di immagini per il controllo degli spazi bianchi**
- **Uso di tabelle per la formattazione della pagina**
- **Scrivere un programma invece di usare l'HTML**



Prima dei CSS

Il risultato?

- **Elevata complessità delle pagine web**
- **Limitata flessibilità**
- **Problemi di interoperabilità**
- **Difficoltà alle persone con disabilità**



I fogli di stile - CSS

- **Lo scopo di un Foglio di Stile è l'aggiunta e la successiva gestione di elementi “presentazionali” ad un documento X-HTML**



I fogli di stile - CSS

- CSS è un “*linguaggio*” relativamente semplice, basato sull'applicazione di regole di formattazione



I fogli di stile - CSS

- **Contenuti**
- **Struttura**
- **Formato**

Documento X-HTML
semanticamente
ben strutturato

– I CSS si basano sull'esistenza di un documento X-HTML correttamente strutturato



I fogli di stile - CSS

Non esiste nessun reale controllo su come una pagina Web apparirà

- **Quindi l'informazione che è contenuta in una pagina web deve essere del tutto separata dall'impaginazione**
 - **Se l'informazione dipende dall'impaginazione, c'è un deficit nell'organizzazione dei contenuti**



Un documento XHTML:

```
<!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"
xml:lang="it">
<head>
  <title>Documento XHTML 1.1</title>
</head>
<body>
  <h1>Un titolo di primo livello</h1>
  <p>Un paragrafo.</p>
  <ul>
    <li>Voce elenco 1</li>
    <li>Voce elenco 2</li>
    <li>Voce <span>elenco 3</span></li>
  </ul>
</body>
</html>
```

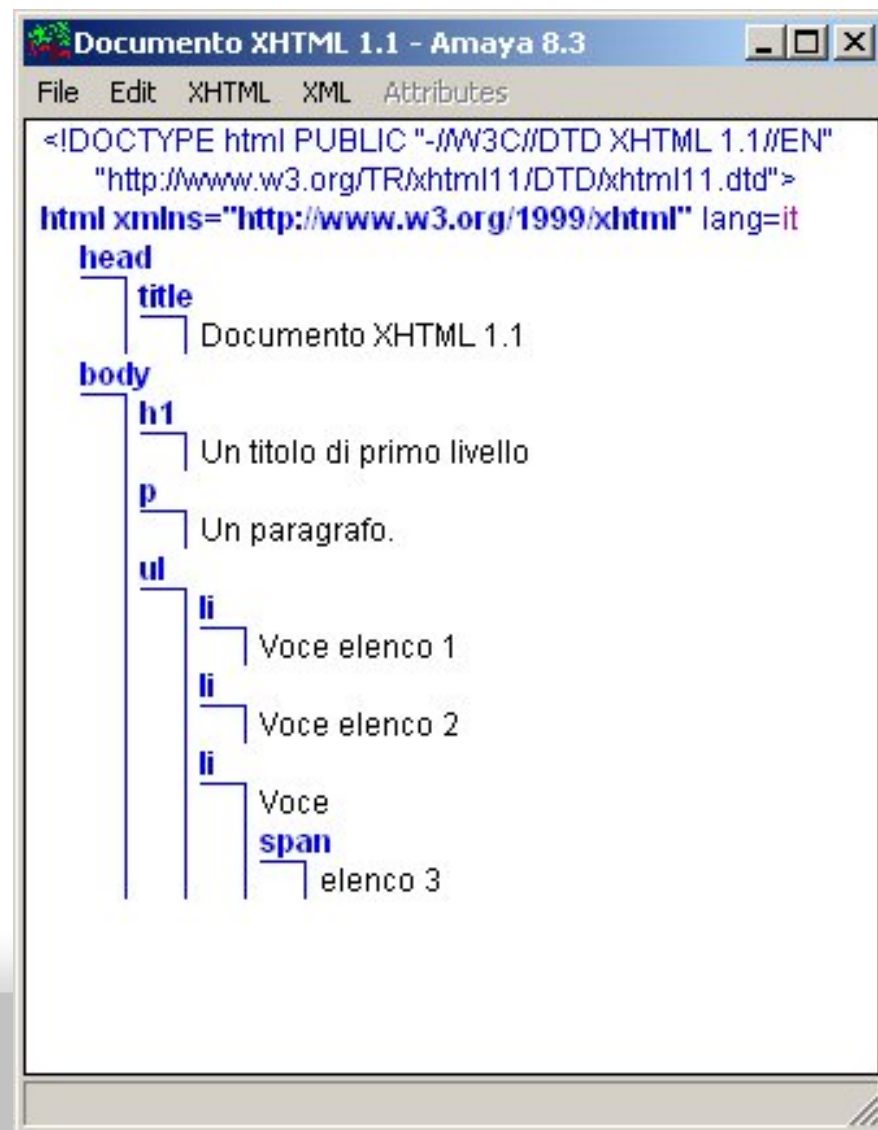
css-doc1.htm



L'albero degli elementi X-HTML

- Un documento X-HTML è rappresentabile con una struttura ad albero

- Esiste un minimo di terminologia da conoscere



L'albero degli elementi X-HTML

- **Genitore (parent)**
 - Ogni elemento dell'albero ha un solo genitore tranne l'elemento radice che ne è privo
- **Figlio (child)**
 - Un elemento A è figlio di un elemento B solo se B è genitore di A
 - *Nell'esempio di prima, P è figlio di BODY ma SPAN non lo è*



L'albero degli elementi X-HTML

- **Discendente (descendant)**
 - Un elemento A è discendente di un elemento B se (1) A è figlio di B o (2) A è figlio di un elemento C che è a sua volta discendente di B.
- **Antenato (ancestor)**
 - Un elemento A è antenato di un elemento B se e solo se B è un discendente di A
- **Fratello (sibling)**
 - Un elemento A è fratello di un elemento B se, e solo se, B ed A condividono lo stesso elemento genitore.



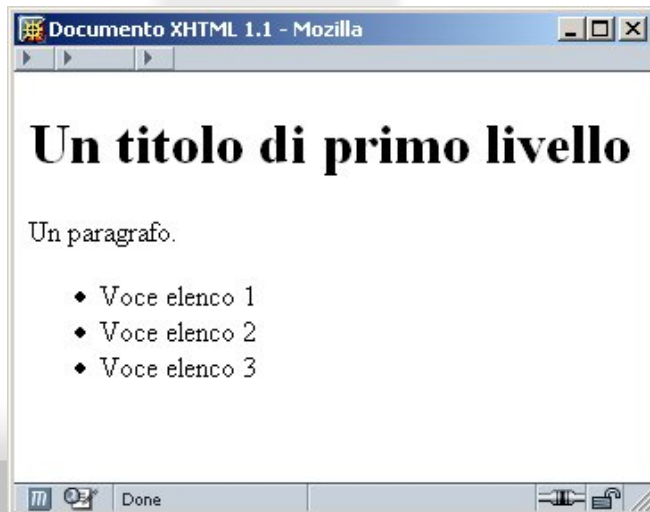
L'albero degli elementi X-HTML

- **Elemento precedente (preceding element)**
 - Un elemento A è precedente di un elemento B solo se A è antenato di B, oppure A è fratello precedente di B
- **Elemento seguente (following element)**
 - Un elemento A è chiamato elemento seguente di un elemento B solo se B è un elemento precedente di A



Foglio di stile di default

- Ogni browser – in assenza di un CSS predisposto dallo sviluppatore - applica ad una pagina X-HTML un foglio di stile di default



- » Per questo alcuni elementi, visualizzati sul browser, assumono un certo formato



Le regole di formattazione

```
selettore { proprietà : valore; }
```

- **Selettore:**
 - E' la parte di documento che sarà influenzata dalla regola
- **Proprietà:**
 - Identifica il particolare aspetto dell'elemento (> 90)
- **Valori**
 - La sintassi dei valori è diversa per ogni proprietà e alcune proprietà ne ammettono più di una



Le regole di formattazione

```
H1 { color: gray; }
```

- **Selettore – Elemento H1**
 - Le proprietà ed i relativi valori impatteranno sull'elemento H1
- **Proprietà - Color**
- **Valori - Grey**
 - Colore grigio del testo relativo all'elemento H1



Le regole di formattazione

```
h1 {  
  color: gray;  
  font-family: Arial,  
  Helvetica, sans-  
  serif;  
  font-size: 18px;  
}
```

**Regola che
interviene su
diverse proprietà
dell'elemento H1**



Le regole di formattazione

```
h1 { color: gray; }  
h2 { color: gray; }  
h3 { color: gray; }
```

In caso di regole
simili su selettori
diversi...



```
h1, h2, h3 { color: gray; }
```

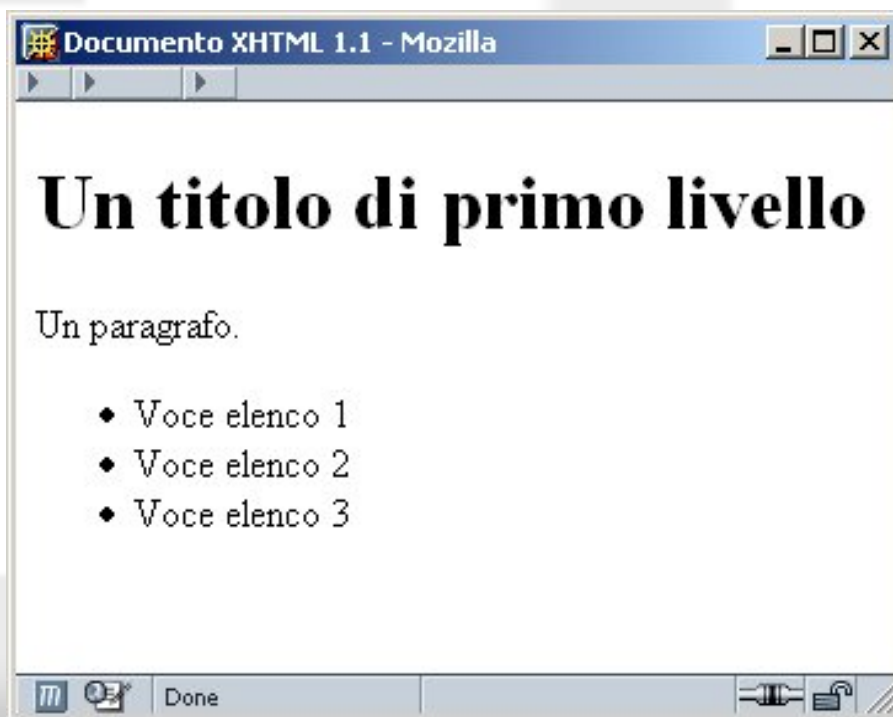


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it">
<head>
<title>Documento XHTML 1.1</title>
<style type="text/css">
  h1 {
    color: gray;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 18px;
  }
</style>
</head>
<body>
  <h1>Un titolo di primo livello</h1>
  <p>Un paragrafo.</p>
  <ul>
    <li>Voce elenco 1</li>
    <li>Voce elenco 2</li>
    <li>Voce <span>elenco 3</span></li>
  </ul>
</body>
</html>
```

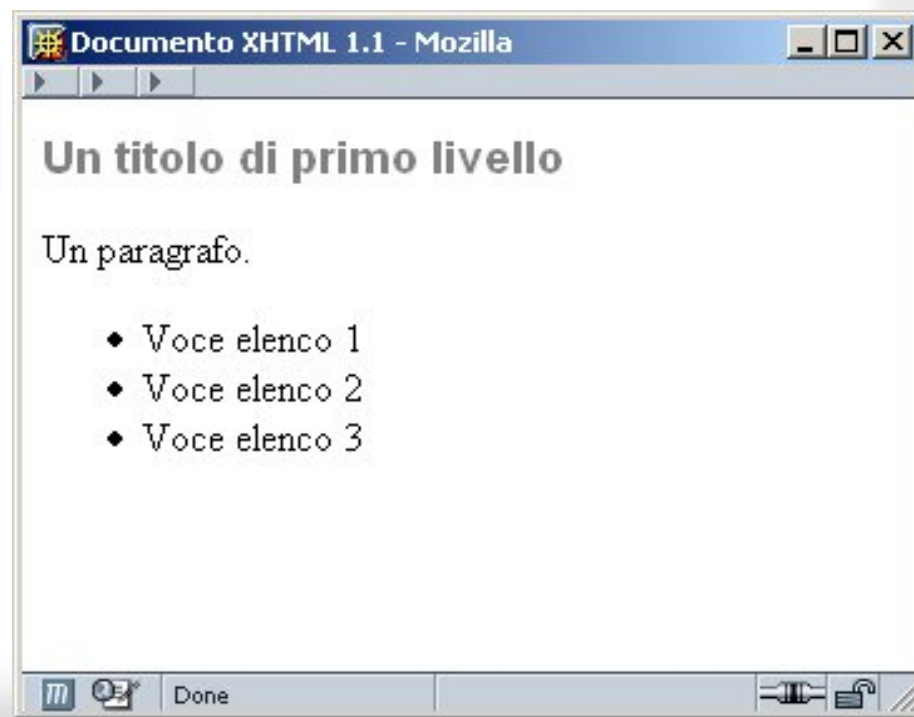


Documento XHTML + CSS

- Il risultato è:



CSS di default



CSS personalizzato



CSS esterni - *Link*

- Un foglio di stile esterno è un file esterno al documento, contenente esclusivamente le regole CSS da applicare
- Esso è associato al documento utilizzando l'elemento HTML *link*, posto all'interno dell'elemento *head*



CSS esterni - *Link*

```
<link rel="stylesheet"
href="http://www.miosito.com/css/stile.css"
type="text/css"
media="screen" />
```

- Il tipo di link, in questo caso stylesheet, foglio di stile
- La posizione del foglio di stile da includere
- Il tipo di foglio di stile collegato: text/css:

- L'attributo **media** si riferisce alla possibilità di applicare fogli di stile a media differenti (per esempio stampa, proiezione, sintesi vocale ecc.)
- I CSS esterni ed incorporati hanno la possibilità di inserire questo attributo



CSS esterni - *@import*

- In alternativa, un foglio di stile esterno può essere importato all'interno di un altro utilizzando la regola **@import**:

```
<style type="text/css">  
@import  
url("http://www.miosito.com/css/stile.css")  
screen;  
</style>
```

- La regola indica la posizione del foglio di stile esterno ed è contenuta all'interno dell'elemento **STYLE**, a sua volta posto in **HEAD**.
- Anche in questo caso può essere indicato il media di destinazione



CSS esterni - *@import*

- La direttiva @import NON è riconosciuta da IE4 e NN4
 - NN4 ha scarse capacità di leggere le istruzioni CSS
 - *Usare @import è una soluzione per evitare che ad NN4 vengano inviati CSS che non è in grado di interpretare*



CSS esterni – *Link* + *@import*

```
<head>
```

```
  <link rel="stylesheet" type="text/css"  
        href=\"css_x_nn4.css\" />
```

```
  <style type="text/css">  
    @import url("css_avanzato.css");  
  </style>
```

```
</head>
```

- Se in `css_avanzato.css` ci sono delle regole presenti anche in `css_x_nn4.css`, quest'ultime vengono sovrascritte



CSS incorporati

- Le regole sono poste all'interno del documento HTML stesso, nell'elemento *style*, a sua volta contenuto da *head*
 - *Nell'esempio di prima abbiamo utilizzato proprio questo metodo*



CSS in linea

- Le dichiarazioni sono direttamente associate all'elemento che influenzano utilizzando l'attributo *style* di HTML:

```
<h1 style="color: gray;">Un titolo  
di primo livello</h1>
```

- È importante notare che nella specifica XHTML 1.1 l'attributo *style* è deprecato



CSS – priorità

- **Priorità in ordine discendente:**
 - **CSS esterno**
 - **CSS incorporato**
 - **CSS in linea**



Selettore universale

- Il selettore universale, indicato da un asterisco *****, ha influsso su ogni elemento nell'albero degli elementi

```
* { color: black; }
```



Selettore di discendente

- Influenza un elemento discendente di un altro elemento
- E' formato da due o più selettori separati da uno spazio, per esempio:

```
li span { background: yellow; }
```

- *Questa regola influenzerà tutti gli elementi SPAN che sono discendenti di “li”*

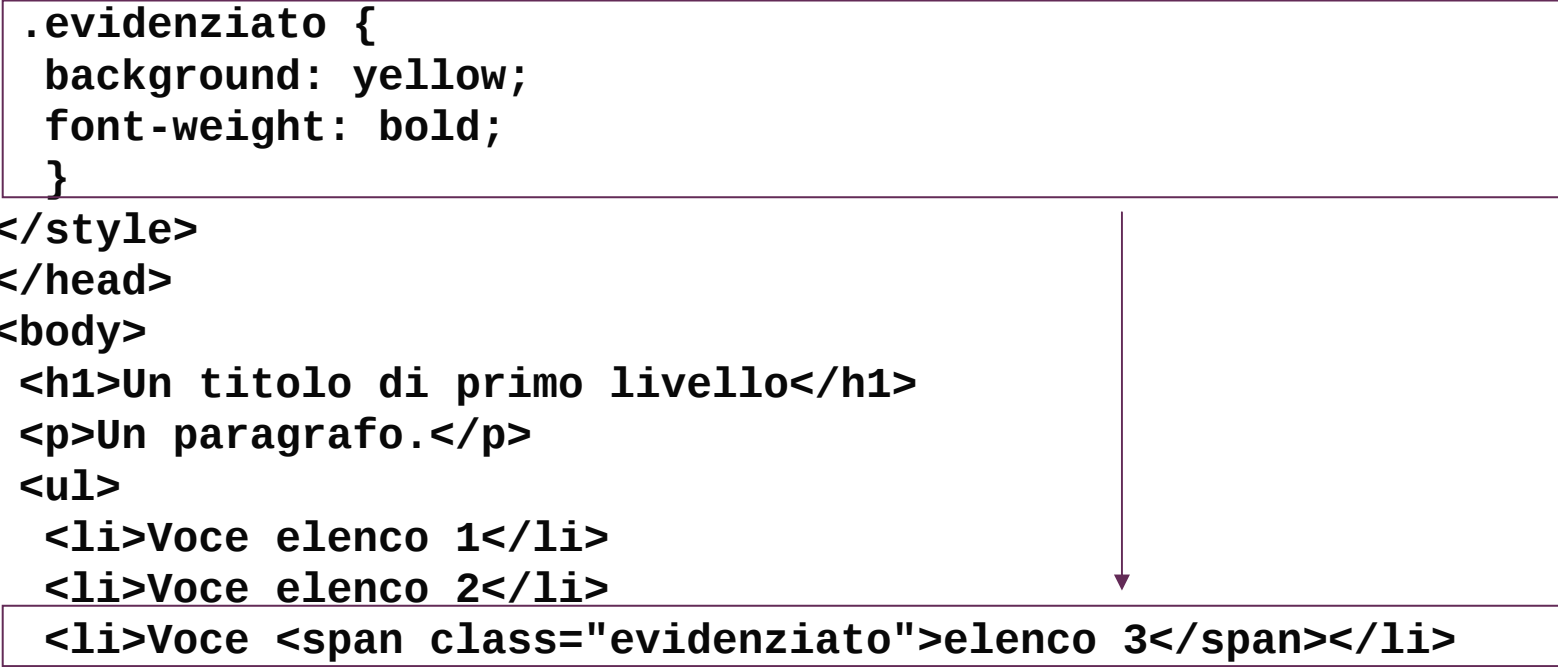


Selettore di classe

- Specifica regole che influenzano elementi con specifici attributi
- I selettori di classe consentono di associare uno stile agli elementi con particolari valori dell'attributo *class* —————→

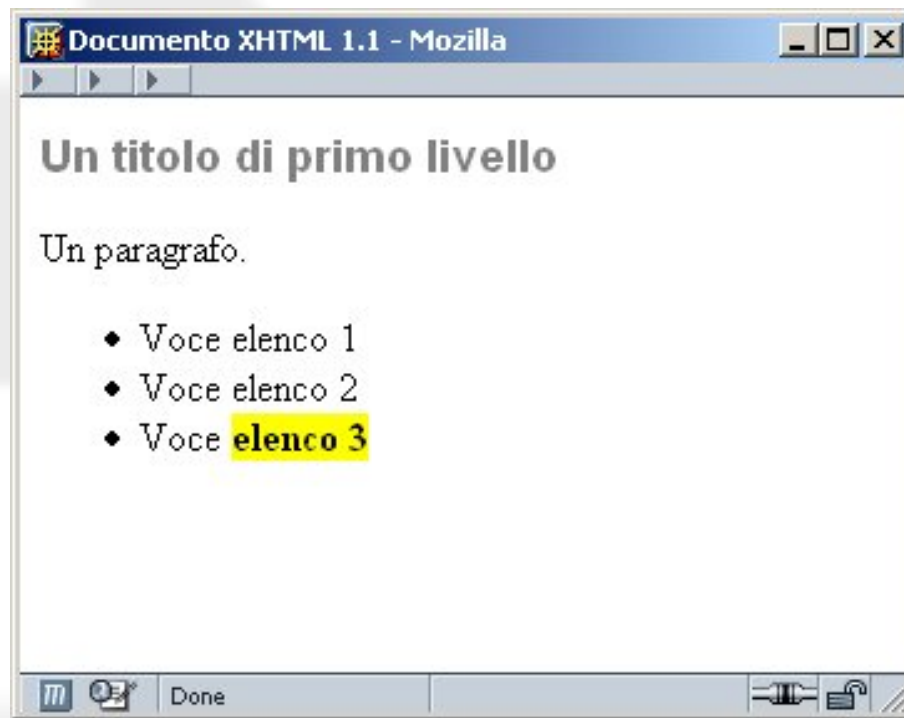


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it">
<head>
<title>Documento XHTML 1.1</title>
<style type="text/css">
  h1 {
    color: gray;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 18px;
  }
  .evidenziato {
    background: yellow;
    font-weight: bold;
  }
</style>
</head>
<body>
  <h1>Un titolo di primo livello</h1>
  <p>Un paragrafo.</p>
  <ul>
    <li>Voce elenco 1</li>
    <li>Voce elenco 2</li>
    <li>Voce <span class="evidenziato">elenco 3</span></li>
  </ul>
</body>
</html>
```



Selettore di classe

- Il risultato è:



Selettore di Id

- I selettori di ID si comportano in modo simile a quelli di classe, ma con un importante differenza:
 - Una classe può essere applicata a più elementi nello stesso documento
 - L'attributo id è univoco, cioè può comparire solo una volta nel documento
 - Questo attributo è utilizzato per identificare in modo univoco l'elemento al quale è associato



Selettore di Id

- Consideriamo questo frammento di codice:

```
<div id="contenuto1">Contenuto del  
primo capitolo</div>
```

- Per associare uno stile particolare a questo titolo di primo livello (*solo a questo*) possiamo impostare una regola in questo modo:

```
#contenuto1 { color: green; }
```



Pseudo-elementi

- Esistono stili tipografici non riproducibili con un normale selettore di tipo, perchè l'HTML del documento non ha elementi preposti a tale scopo
- CSS permette di farlo utilizzando gli **pseudo-elementi**
- Consideriamo, per esempio, un **capolettera**
 - *HTML non ha un tag per il capolettera, a meno che non si racchiudano tutte le prime lettere di ciascun paragrafo all'interno di uno SPAN associandogli una classe, ma sarebbe un lavoro lungo e noioso con documenti lunghi*



Pseudo-elementi

:First-letter

```
p:first-letter { font-size: 24px;  
font-weight: bold; }
```

- Il primo carattere di tutti i paragrafi del documento è dimensionato a 24 pixel e in grassetto

css-doc4-htm



Pseudo-elementi: altri

:First-line

- Influenza la prima riga dell'elemento

:before

- Inserisce del contenuto generato dal foglio di stile prima del contenuto dell'elemento

:after

- Inserisce del contenuto generato dal foglio di stile dopo del contenuto dell'elemento



Pseudo-classi

- **Classificano gli elementi utilizzando caratteristiche diverse dal nome, gli attributi o il contenuto**
 - Possono essere dinamiche, nel senso che un elemento può acquisire o perdere pseudo-classi, mentre l'utente interagisce con il documento



Pseudo-classi

```
<a href="http://www.webaccessibile.org/">Webaccessibile</a>
```

a:link { color: blue; }

- Modifica l'aspetto di un collegamento non ancora visitato

a:visited { color: gray; }

- Modifica l'aspetto di un collegamento visitato

a:hover { color: orange; }

- Si applica quando l'utente posiziona il puntatore del mouse sul box generato dall'elemento

a:active { color: red; }

- Si applica quando l'elemento viene attivato dall'utente, cioè nell'intervallo di tempo tra la pressione del pulsante del mouse e il suo rilascio



Ereditarietà

- *I discendenti di un elemento ereditano i valori delle proprietà associate all'antenato...*

...a meno che non sia loro applicata una regola più specifica



Ereditarietà

```
<body>
  <h1>Titolo</h1>
  <p>Paragrafo.</p>
  <h2>Sottotitolo</h2>
  <p>Paragrafo.</p>
</body>
```

X-HTML

- **body { color: black; }**
 - Tutto il testo è nero
- **body { color: black; }**
p { color: red; }
 - Tutto il testo è nero tranne quello all'interno degli elementi *p* e dei loro discendenti

CSS (2 esempi)



Ereditarietà

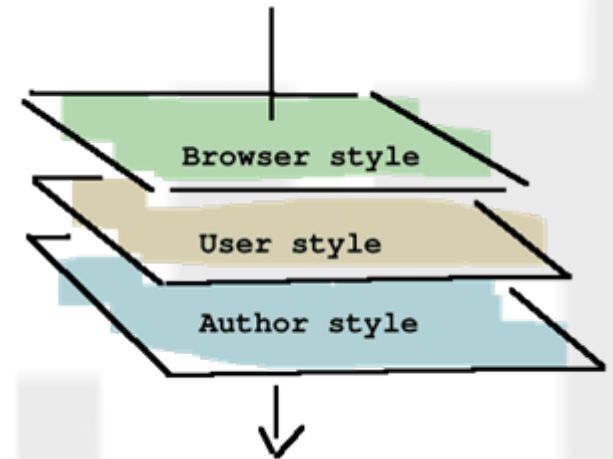
- Non tutte le proprietà sono ereditate
- Quelle ereditabili non sono tali per tutti gli elementi
 - Il valore *inherit* da la possibilità di decidere se il valore di una data proprietà sarà ereditato anche se di norma non lo sarebbe



Cascata

I fogli di stile assegnati a un documento possono avere tre origini:

- **Lo user agent (il browser)**
- **L'utente**
- **L'autore**
 - **Per esempio il CSS di default applicato ai documenti in assenza di altri fogli di stile**



Cascata

- **In presenza di diversi fogli di stile potrebbero esserci delle regole in conflitto fra loro**
 - **E' previsto un ordine a cascata, con delle priorità ben stabilite**
 - **Ogni regola ha un peso, e le regole con peso maggiore hanno la precedenza sulle altre**



Cascata - *!important*

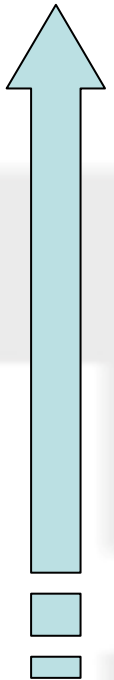
- Il CSS dell'autore ha un peso maggiore di quello dell'utente, tranne che per le dichiarazioni precedute dal costrutto *!important*
- Questo impone loro precedenza nell'ordine di cascata
 - Le dichiarazioni presenti nel foglio di stile predefinito dal browser hanno il peso minore



Cascata – le priorità

Ecco lo schema in ordine ascendente:

- 1. Foglio di stile predefinito del browser**
- 2. Foglio di stile normale dell'utente**
- 3. Foglio di stile normale dell'autore**
- 4. Foglio di stile importante dell'autore**
- 5. Foglio di stile importante dell'utente**



Cascata

- **Successivamente, il browser ordina le regole in funzione della specificità del selettore**
- **I selettori maggiormente specifici avranno la precedenza su quelli più generali**

Ma cosa si intende con il termine specificità?



Cascata – calcolare la specificità

Ci si basa sulla composizione del selettore, con una sequenza di quattro numeri separati da virgola (a, b, c, d), ottenuti in questo modo:

- a. =1 se il selettore è un attributo style di HTML piuttosto che un selettore vero e proprio, altrimenti 0 (= a);
- b. = il numero di attributi ID nel selettore (= b);
- c. = il numero di altri attributi o pseudo-classi nel selettore (= c);
- d. = il numero di nomi di elementi o pseudo-elementi nel selettore (= d);



Cascata – calcolare la specificità: es.

```
ul li { color: black; background: white; }
```

- $a = 0, b = 0, c = 0, d = 2$ poiché nel nostro selettore compaiono solamente due nomi di elementi
- Quindi la specificità è 0,0,0,2



Cascata – calcolare la specificità: es.

```
ul li { color: black; background: white; }
```

regola 1, specificità : 0,0,0,2

```
ul li.evidenziato { background: yellow; }
```

regola 2, specificità : 0,0,1,2



Cascata – calcolare la specificità: es.

- La seconda regola ha una specificità maggiore:
 - Essa influenzerà solo gli elementi *LI* discendenti di *UL* ai quali è associata la classe *evidenziato*
- Nell'ordine di cascata essa avrà la precedenza, indipendentemente da dove compaia foglio di stile



Cascata – esempio di specificità

- Se però, nel documento c'è un foglio di stile in linea, esso avrà la precedenza su qualsiasi altra dichiarazione

Css-doc6.html



II Box Model

- Una pagina X-HTML è un insieme di box rettangolari
 - Questo vale sia per gli elementi di **blocco** che per quelli **inline**
- **BOX MODEL:**
 - Insieme di regole relative all'aspetto degli elementi di blocco e al loro mutuo rapporto



Il Box Model

- Ciascun box ha degli elementi costitutivi, modificabili con i CSS:



Box Model: Content

- E' la zona centrale del box, in cui si trova il contenuto (testo, immagini...)
- Le dimensioni della zona contenuto si gestiscono con la proprietà **width** (lunghezza) e **height** (altezza)



Box Model: Padding

- E' uno spazio vuoto tra il contenuto e il bordo dell'elemento
 - *Un eventuale colore di sfondo per un elemento si estende dall'area del contenuto alla zona di padding*



Box Model: Padding

5 parametri:

- **Padding** 
- **Padding-top**
- **Padding-bottom**
- **Padding-left**
- **Padding-right**



Box Model: parametro *padding*

- E' una proprietà a sintassi abbreviata
- Si possono specificare i valori per tutti i lati di un elemento
- E' applicabile a tutti gli elementi e non è ereditata



Box Model: parametro *padding*

Ammette diversi tipi di valori:

- **Numerico con unità di misura**
- **Percentuale.**
- ***Auto***
 - In questo caso la distanza sarà calcolata rispetto alla larghezza dell'elemento contenitore



Box Model: parametro *padding*

```
div {padding: 15px 15px 10px 25px;}
```

- L'ordine di lettura va inteso in senso orario
- In pratica usare la sintassi di sopra, equivale a scrivere:

```
div {  
  padding-top: 15px;  
  padding-right: 15px;  
  padding-bottom: 10px;  
  padding-left: 25px; }
```



Box Model: parametro *padding*

Un'altra abbreviazione si può avere usando tre, due o un solo valore:

- Tre valori:
 - Il primo si riferisce al margine superiore, il secondo a quelli sinistro e destro, il terzo a quello inferiore
- Due valori:
 - Il primo si riferisce ai lati superiore e inferiore, il secondo al sinistro e al destro
- Un solo valore
 - Uguale distanza sui quattro lati



Box Model: Border

- E' una linea, dai parametri (stile e colore) variabili, che circonda la zona del padding ed il contenuto



Box Model: Border

- La definizione accurata dei bordi può risultare complessa per il numero di proprietà coinvolte
 - Massima flessibilità
 - Complicata gestione del codice



Box Model: Border

- **Sintassi a proprietà singole**

```
- selettore {  
  border-<lato>-color: <valore>;  
  border-<lato>-style: <valore>;  
  border-<lato>-width: <valore>;  
}
```

- **Sintassi compatta**

```
- selettore { border-<lato>:  
  <valore width> <valore style>  
  <valore color>; }
```



Box Model: Border *style*

- **none**
 - Nessun bordo, nessuno spessore
- **hidden**
 - Equivalente a none
- **dotted**
- **dashed**
- **solid**
- **double**
- **groove**
- **ridge**
- **inset**
- **outset**



Box Model: Border *width* (spessore)

Può assumere i seguenti valori:

- Numerico con unità di misura**
- Thin - Bordo sottile**
- Medium - Bordo di medio spessore**
- Thick – Bordo largo**



Box Model: Border (*abbreviato*)

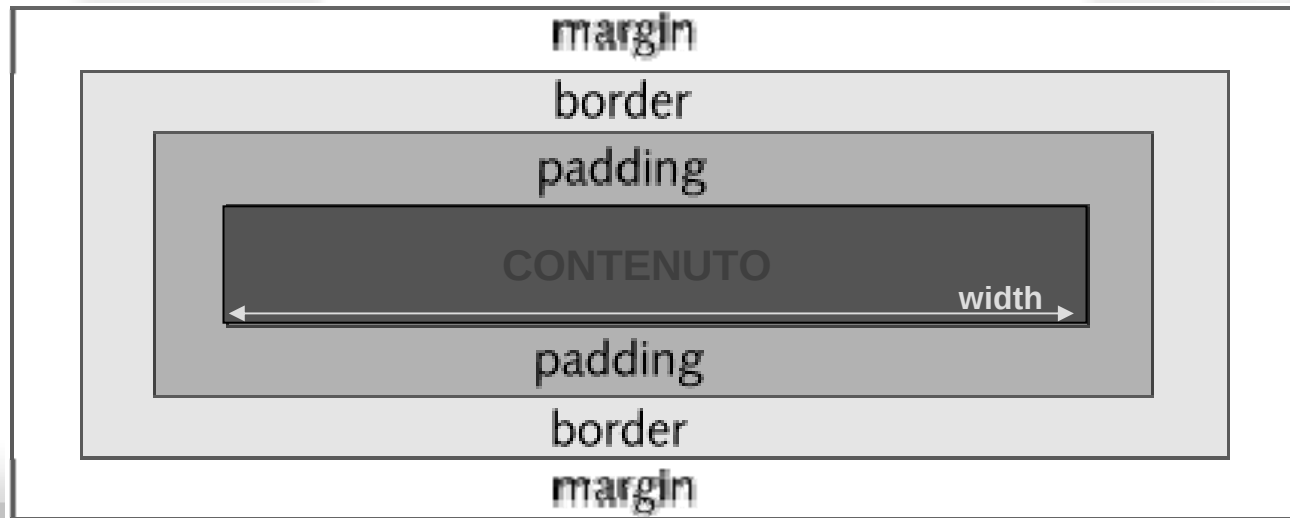
```
selettore {border: <valore spessore>  
<valore stile> <valore colore>;}
```

- In questo caso basta un'unica regola a settare le impostazioni per i quattro bordi
 - Il suo uso è limitato a un solo caso molto comune: che i quattro bordi abbiano tutti lo stesso colore, lo stesso stile e lo stesso spessore



Box Model: Margin

- E' lo spazio di dimensioni variabili che separa un elemento da quelli adiacenti
 - *E' l'unico parametro che può avere valori negativi*



Box Model: Margin

5 parametri:

- **Margin**
- **Margin-top**
- **Margin-bottom**
- **Margin-left**
- **Margin-right**



Box Model: Margin

- Anche per margin vale quanto detto per padding:
 - Sintassi compatta
 - Ordine di lettura dei valori
 - Eventuali abbreviazioni



Box Model: larghezza del box

- Larghezza del content = *width*
- Larghezza effettiva di un box:

*margin sn + border sn + padding sn +
content +
padding dx + border dx + margin dx*



Box Model: larghezza del box

- In assenza del parametro *width*, il box assumerà la stessa larghezza del *content* del box che lo contiene



Box Model: margini verticali

- Per due box uno sull'altro che hanno un margine inferiore e uno superiore la distanza non sarà data dalla somma delle due distanze
- A prevalere sarà invece la distanza maggiore tra le due (**margin collapsing**)
 - Tale meccanismo non si applica ai box adiacenti in senso orizzontale



Box Model: IE 5 win

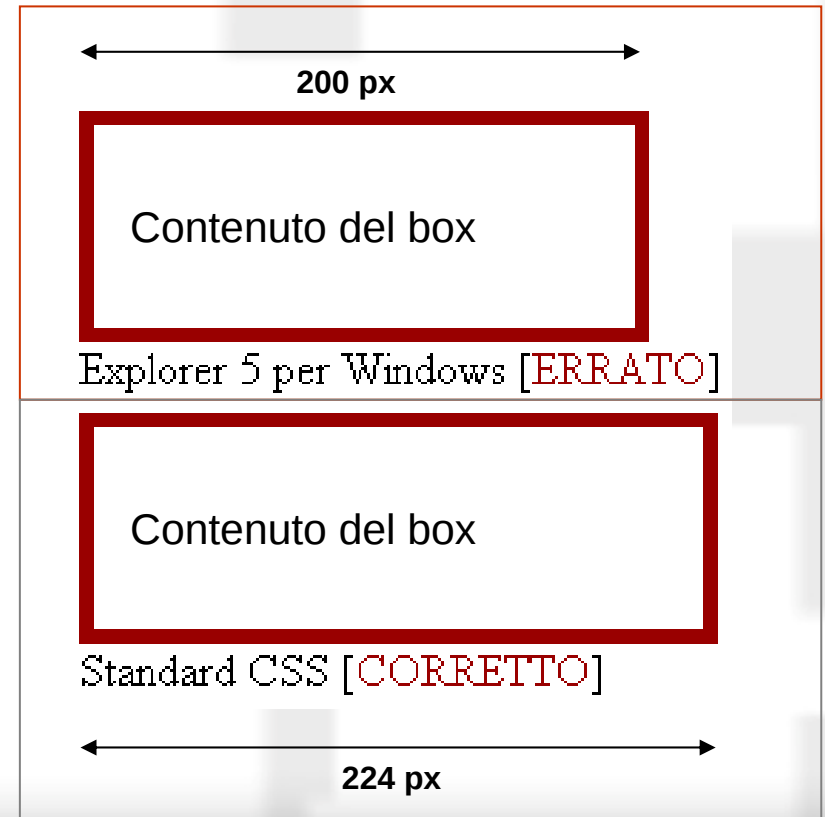
IE5 per Windows ha una particolarità...

- **Width:**
 - Larghezza del box fino al bordo



Box Model: IE 5 win

```
<style type="text/css">
<!--
.box {
width:200px;
border:5px solid #900;
padding:7px;
margin:10px
}
-->
</style>
</head>
<body>
<div class="box">
Contenuto del box
</div>...
```



Box Model: IE 5 win

Esistono due soluzioni:

- 1. Tecnica di TanteK**
- 2. Creare una classe apposita per la specifica della larghezza**



IE 5 win – 1: tecnica di Tantek

```
<style
type="text/css">
<!--
.box {
width:224px;
border:5px solid #900;
padding:7px;
margin:10px
voice-family: "\"}\"";
voice-family: inherit;
width:200px;
}
-->
</style>
```

1. Si specifica width con la larghezza totale (224)
2. Poi si fornisce la larghezza corretta (200) con un comando che IE5 non è in grado di interpretare per colpa di un bug



IE 5 win – 1: tecnica di Tantek

```
<style
type="text/css">
<!--
.box {
width:224px;
border:5px solid #900;
padding:7px;
margin:10px
voice-family: "\"}\"\"";
voice-family: inherit;
width:200px;
}
-->
</style>
```

- Quando IE5 incontra l'istruzione ***voice-family:*** ***"\"}\"\"";***, un bug blocca l'interpretazione delle regole fino a }
- L'istruzione ***voice-family:inherit;***, ripristina il giusto valore per la proprietà ***voice-family***
- Gli altri browser continuano il parsing delle regole assegnate alla class ***' .box '***, sovrascrivendo la larghezza assegnata in precedenza



IE 5 win – 1: tecnica di Tantek

- Anche Opera 5 ha lo stesso bug di IE5
- Però Opera 5 ha il model box in linea con le specifiche CSS

```
html>body .box { /* "be nice to Opera 5" !!  
non lasciare spazi prima o dopo '>' !! */  
color:blue;  
}
```

permette ad Opera 5 di leggere la giusta istruzione.



IE 5 win – 2: classe apposita

- *Se nel CSS si imposta solo la larghezza, lasciando a 0px il bordo e il padding...*
- .. non c'è diversa interpretazione fra IE5/Win e un browser che interpreti con correttezza il box model.
 - Questo può suggerire un “trucco” per creare box in maniera crossbrowser



IE 5 win – 2: classe apposita

```
<style type="text/css">
<!--
.box {
width:224px;
}
.content {
border:5px solid #900;
padding:7px;
}
-->
</style>
...
<div class="box">
<div class="content">
Contenuto del box
</div>
</div>
```

- Nella classe *'content'* vengono specificati il padding e il bordo, nella classe *'box'* viene specificata la larghezza dei contenuti
- In questo modo, il contenuto sarà di 200px su IE5/Win ed anche sugli altri browser



Posizionare gli elementi

- Una volta creata la struttura di una pagina X-HTML, servono gli strumenti adatti per assegnare a ciascuno di essi il formato e la posizione sulla pagina che verrà visualizzata dal browser



Proprietà *Display*

- Permette la definizione ed il trattamento della presentazione di un elemento
 - Con display è possibile stravolgere il comportamento di default di qualsiasi elemento
 - E' una proprietà ereditata



Proprietà *Display*

- **Immagine**
 - E' un elemento *Inline*
 - La proprietà *Display* può renderla elemento di *blocco*
- **Paragrafo**
 - E' il più classico degli elementi di blocco
 - La proprietà *Display* può renderlo un elemento di linea



Proprietà *Display*

```
<selettore> {display: <valore>;}
```

- I possibili valori sono tantissimi
- Solo alcuni di essi risultano ben supportati...



Proprietà *Display* : i valori

- **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.
 - Il valore è supportato solo da Opera 5/6 e succ. e parzialmente da Explorer 5 Mac. e succ.



Proprietà *Display* : i valori

- **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
- Di fatto non è presente nel documento, e non genera alcun box
 - Diversa è la proprietà **visibility:hidden** che invece si limita a nascondere l'elemento



Proprietà *Float*

- ***Float*** rimuove un elemento dal normale flusso del documento e lo sposta su uno dei lati del suo elemento contenitore
 - Il contenuto che circonda l'elemento scorrerà intorno ad esso sul lato opposto rispetto a quello indicato come valore di *float*
 - *La proprietà non è ereditata*



Proprietà *Float*

```
<selettore> {float: <valore>;}
```

I possibili valori:

- **Left**
 - L'elemento viene spostato sul lato sinistro del box contenitore, il contenuto scorre a destra
- **Right**
 - L'elemento viene spostato a destra, il contenuto scorre a sinistra
- **None**
 - Valore di default in mancanza di una dichiarazione esplicita, l'elemento resta dov'è



Proprietà *Float*

- **Attenzione:**
 - Float con le immagini non da problemi perchè esse hanno una dimensione intrinseca che il browser riconosce
 - *Con tutti gli altri elementi si deve esplicitamente impostare una dimensione orizzontale con la proprietà **width***

[css-doc13.htm](#)

[css-doc14.htm](#)

[css-doc15.htm](#)



Proprietà *Clear*

- La proprietà ***clear*** impedisce che accanto ad un elemento possano comparirne altri con il *float*
 - *Si applica solo agli elementi di tipo blocco e non è ereditata*



Proprietà *Clear*

- In pratica, se il *float* può spostare un elemento dal flusso normale del documento, esso potrebbe venirsi a trovarsi in posizioni non desiderate
 - *Per esempio al fianco di altri elementi che vogliamo invece tenere separati*
- ***Clear*** risolve questo problema



Proprietà *Position*

- ***Position*** è la proprietà più importante per operare sulla posizione degli elementi
- Esso specifica anche la loro modalità di presentazione sulla pagina
 - *Si applica a tutti gli elementi e non è ereditata*



Proprietà *Position*

```
<selettore> {position: <valore>;}
```

Possibili valori:

- **static**
- **absolute**
- **fixed**
- **relative**



Proprietà *Position: static*

- E' il valore di default per tutti gli elementi non posizionati secondo un altro metodo
- Rappresenta la posizione normale che ciascun elemento occupa nel flusso del documento



Proprietà *Position: absolute*

- Il box dell'elemento viene rimosso dal flusso del documento e posizionato secondo le coordinate fornite con le proprietà *top*, *left*, *right* o *bottom*



Proprietà *Position: absolute*

- Il posizionamento avviene in riferimento al **box contenitore** dell'elemento
- Questo è il primo elemento antenato (**ancestor**) avente un posizionamento diverso da **static**



Proprietà *Position: absolute*

- Se tale elemento non esiste il posizionamento avviene in base all'elemento radice **HTML**
 - *Cioè l'area del browser che contiene il documento e che ha l'origine nell'angolo superiore sinistro*
- Un elemento posizionato in modo assoluto scorre insieme al resto del documento



Proprietà *Position: fixed*

- Anche con *Fixed* il box dell'elemento viene tolto dal flusso del documento
- Però, per *fixed* il contenitore è sempre il cosiddetto *viewport*
 - Con questo termine si intende la finestra principale del browser



Proprietà *Position: fixed*

- Un box posizionato con *fixed* non scorre con il resto del documento ma rimane fisso al suo posto
 - *Purtroppo, il valore fixed non è supportato da Explorer su Windows*



Proprietà Position: relative

- L'elemento è posizionato relativamente al suo box contenitore
- Il contenitore è il posto che l'elemento avrebbe occupato nel normale flusso del documento



Proprietà *Position: relative*

- La posizione viene impostata con le proprietà **top, left, bottom, right**
- Ma qui le coordinate non indicano un punto fisso, ma lo spostamento rispetto al box contenitore



Proprietà *visibility*

```
<selettore> {visibility: <valore>;}
```

- Stabilisce se un elemento risulta visibile o nascosto
 - *Si applica a tutti gli elementi e non è ereditata*



Proprietà *visibility*

Valori possibili

- **Visible**
 - L'elemento è visibile (default)
- **Hidden**
 - L'elemento è nascosto, ma resta al suo posto nel layout (apparirà una zona vuota)
 - In ciò è diverso dal **none** di **display**



Proprietà *Z-index*

```
<selettore> {z-index: <valore>;}
```

- Si imposta l'ordine di posizionamento dei vari elementi sulla base di una scala di livelli.
- E' possibile che un elemento si sovrapponga ad un altro rendendolo (in parte o del tutto) illeggibile



CSS Techniques for WCAG 1.0

- **Si tratta di un documento del W3C che descrive una serie di tecniche e suggerimenti per sviluppare fogli di stile a cascata conformi alle WCAG 1.0**
 - 17 sezioni**



1. Meno manutenzione, più coerenza

A. Utilizzare un numero minimo di fogli di stile

- Si riduce drasticamente il tempo necessario alla manutenzione
- È chiaro che minore sarà il numero di fogli di stile minore sarà il tempo e il lavoro necessario ad effettuare modifiche



1. Meno manutenzione, più coerenza

B. Utilizzare CSS esterni piuttosto che incorporati ed evitare l'utilizzo di CSS in linea

- Un CSS esterno fornisce da solo il completo controllo sull'aspetto di un numero potenzialmente infinito di documenti
- Al contrario, l'uso di stili in linea, in un certo senso, snatura la funzione dei fogli di stile, legando informazioni presentazionali al singolo elemento HTML.
- La presenza di CSS in linea rende il codice confuso



1. Meno manutenzione, più coerenza

C. In tutti i fogli di stile utilizzare lo stesso nome per la definizione di classi che rappresentano lo stesso concetto



2. Imposizione dello stile dell'utente

- La possibilità da parte dell'utente di **controllare lo stile di presentazione** dei documenti, ne incrementa notevolmente l'accessibilità
- Un utente con problemi di ipovisione potrebbe:
 - Imporre al browser di visualizzare caratteri con dimensioni maggiori di quelle definite dal CSS dell'autore
 - Incrementare il contrasto tra il colore del testo e quello dello sfondo
 - ...



2. Imposizione dello stile dell'utente

- Il documento del W3C prosegue indicando, browser per browser, la procedura per applicare un foglio di stile personalizzato
 - *Questi dettagli esulano dagli scopi del nostro corso*



3. Unità di misura

- **Alcune proprietà CSS contengono valori che rappresentano misurazioni di lunghezze**
- **Il valore numerico della lunghezza è subito seguito dall'abbreviazione dell'unità di misura scelta, con l'unica eccezione del valore 0**



3. Unità di misura relative

- **Esprimono lunghezze calcolate in relazione ad altre**

ABCABCABC
ABCABCABC
ABCABCABC



3. Unità di misura relative

```
p {  
  font-size: 10px;  
  margin-bottom: 1.2em;  
}
```

- Margin-bottom è impostata con l'unità di misura relativa **em**
- Il valore di questa unità è calcolato in relazione a quello della proprietà font-size dell'elemento nel quale è utilizzata



3. Unità di misura relative

```
p {  
  font-size: 10px;  
  margin-bottom: 1.2em;  
}
```

- Nella nostra regola il valore del margine inferiore è **1.2em**, 1,2 volte le dimensioni del carattere dell'elemento e cioè **12 pixel**



3. Unità di misura relative

```
p { font-size: 10px; }  
span { font-size: 1.2em; }
```

- Quando la proprietà font-size è espressa in em, *il suo valore si relaziona a quello delle dimensioni del carattere dell'elemento genitore*
- In questo caso le dimensioni del carattere dell'elemento *SPAN* saranno 1,2 volte quelle dell'elemento genitore *P*



3. Unità di misura relative

- EM
 - Il nome di questa unità di misura traeva origine dalle dimensioni in larghezza della lettera M, anche se attualmente *si riferisce all'altezza del corpo di un carattere*



3. Unità di misura relative: il Pixel

- L'unità di misura pixel (px) è considerata **relativa** perchè dipende dalla risoluzione del dispositivo di visualizzazione
- E' necessario pensare ad essa come ad *un'unità logica dipendente dal dispositivo*, piuttosto che fisica



3. Unità di misura relative: il Pixel

- **Schermo alla massima risoluzione:**
 - La dimensione fisica di un pixel sarà identica alla grandezza del punto più piccolo che lo schermo può rappresentare
- **Risoluzione a un valore intermedio:**
 - Un pixel sarà più "largo" della dimensione fisica di un punto sullo schermo e corrisponderà a un insieme di essi



3. Unità di misura relative: il Pixel

- Utilizzare i pixel per i valori della proprietà *font-size* è **sconsigliabile**
- In questo caso l'utente non avrà la possibilità di ridimensionare il testo
 - *Per esempio utilizzando la funzione di ingrandimento del carattere di IE*



3. Unità di misura percentuali

- Le dimensioni definite in percentuale (%) sono sempre relative a un altro valore
- Nella specifica del linguaggio, per ogni proprietà che ammette valori in percentuale, sono definiti anche i valori ai quali le percentuali si riferiscono
 - Per esempio, per la proprietà *line-height*, un valore in percentuale si riferisce alla proprietà *font-size* dell'elemento stesso



3. Unità di misura assolute

Unità	Corrisponde a
in (Pollici)	1 pollice = 2.54 cm
cm (Centimetri)	1 cm = 0,394 pollici
mm (Millimetri)	mm = 1 cm, 1 cm = 0.0394 pollici, 25.4 mm = 1 pollice
pt (Punti)	1 punto = 1/72 pollici oppure 72 punti = 1 pollice
pc (Pica)	1 pica = 12 punti, 6 pica = 1 pollice



3. Unità di misura: CSS Tech1

- Utilizzare l'unità *em* per il dimensionamento dei caratteri
- Utilizzare unità di misura *relative* e *percentuali* anche per il posizionamento assoluto
 - Così si può, per esempio, posizionare un'immagine distanziata di 3em dal limite superiore (top) del suo elemento contenitore
 - Questa è una distanza relativa alle dimensioni del carattere correntemente utilizzato



3. Unità di misura: CSS Tech1

- **Utilizzare unità di misura assolute solo quando le caratteristiche fisiche dell'output sono conosciute, come nelle immagini bitmap**



4. Contenuto generato

Con i CSS è possibile generare contenuto non appartenente al DOM del documento

- **Gli elementi grafici di un elenco puntato**
- **La numerazione progressiva di un elenco numerato**
 - *Non sono presenti nel codice, ma sono generati dal browser*



4. Contenuto generato: CSS Tech1

- Fornire un **equivalente testuale** per ogni immagine importante o testo generato dai fogli di stile
 - per esempio con le proprietà *background-image*, *list-style* o *content*
- Assicurarsi che il contenuto importante appaia nel documento
 - Il testo generato dai fogli di stile non è parte del codice XHTML e, quindi, non sarà accessibile alle tecnologie assistive che accedono al documento tramite il DOM



5. Caratteri

- Il primo consiglio è indicare sempre anche una delle famiglie generiche:

Nome	Tipologia
<i>serif</i>	Con grazie, per esempio Times New Roman
<i>sans-serif</i>	Senza grazie, per esempio Arial
<i>cursive</i>	Caratteri che somigliano alla scrittura calligrafica
<i>fantasy</i>	caratteri "fantasiosi" non ascrivibili ad altre categorie, per esempio Comic Sans o Western
<i>monospace</i>	caratteri nei quali le lettere occupano tutte lo stesso spazio orizzontale, per esempio Courier o Andale Mono



5. Caratteri

```
body { font-family: Arial, Helvetica, sans-serif; }
```

- Se Arial e Helvetica non fossero presenti nel sistema, potremo essere ragionevolmente certi che il documento sarà visualizzato dal browser con un carattere senza grazie



6. Effetti di stile del testo

Possiamo utilizzare le proprietà :

- *Text-transform* per il formato del testo maiuscolo, minuscolo e con iniziali maiuscole
- *Text-decoration* per il testo sottolineato, sopra-lineato o lampeggiante
 - Attenzione al lampeggiamento del testo



7. Testo al posto delle immagini

- **Se si ritiene indispensabile utilizzare immagini per creare effetti testuali (come caratteri speciali, trasformazioni, ombreggiature ecc.) l'immagine dovrà essere resa accessibile inserendo testo equivalente con l'attributo *alt***



8. Formato e posizionamento del testo

Si possono usare le seguenti proprietà:

- *Text-indent* per il rientro
 - Non utilizzare BLOCKQUOTE o qualsiasi altro elemento strutturale per il rientro del testo
- *Letter-spacing* e *word-spacing*
 - Per modificare la spaziatura delle lettere e delle parole



8. Formato e posizionamento del testo

- **White-space**, avente i seguenti valori:
 - **Normal** (default)
 - Impone di ignorare lo spazio bianco nel testo presente nel codice, collassando tabulazioni, spazi multipli e interruzioni di riga.
 - **Pre**
 - Il browser rispetta gli spazi bianchi, comprese le interruzioni di riga. Il testo sarà visualizzato esattamente come appare nel codice della pagina.
 - **Nowrap**
 - Come normal ma elimina le interruzioni di riga. Il testo apparirà, quindi, tutto su una sola linea



8. Formato e posizionamento del testo

- ***Direction* e *unicode-bidi*** per controllare la direzione del testo.
- ***:first-letter* e *:first-line***
 - Pseudoelementi utili per impostare stili che si applicano alla prima lettera o alla prima riga del testo di un elemento



9. Colori: contrasto

- **È necessario assicurarsi che il colore in primo piano ed il colore di sfondo siano ben contrastati**
 - I vantaggi dell'elevato contrasto sono evidenti per un utente ipovedente o con deficit legati alla visualizzazione dei colori



9. Colori: contrasto

- **Juicy Studio: Colour Contrast Analyzer**
 - Utilizza l'algoritmo per il contrasto del colore del W3C per testare i valori contenuti nel nostro foglio di stile.
 - La traduzione italiana, perfettamente funzionante, del test si trova nella sezione CSS di Webaccessibile.org

<http://www.webaccessibile.org/argomenti/argomento.asp?cat=530>



9. Colori

- **Non affidare le informazioni al solo colore**
 - Per richiedere un input dall'utente, non scrivere "Per favore seleziona una voce tra quelle elencate in verde".
- **Le informazioni devono essere disponibili anche attraverso altri effetti di stile ed il contesto**



10. Tracce contestuali negli elenchi

- **Se un elenco è ben codificato, i fogli di stile possono essere impiegati per fornire agli utenti tracce semantiche contestuali**



10. Tracce contestuali negli elenchi

- NB: un doveroso appunto va fatto per l'utilizzo della proprietà `display: none;`
- Alcuni screen reader applicano questa proprietà, non leggendo il contenuto degli elementi ai quali è associata



11. Layout, posizionamento e livelli

- Layout, posizionamento, livelli e allineamento dovrebbero essere impostati utilizzando i fogli di stile
- In particolare con la proprietà *float* e il posizionamento assoluto



11. Layout, posizionamento e livelli

- **Gli autori dovrebbero sempre progettare documenti che mantengano senso anche senza i fogli di stile**



12. Righe e bordi

- **Righe e bordi possono trasmettere la nozione di "separazione" agli utenti in grado di vedere**
- **Questo significato, però, non può essere riportato in un contesto non visivo**



12. Righe e bordi

- Se una riga creata con l'elemento *HR* è utilizzata con un significato strutturale, bisogna assicurarsi di indicare la struttura anche in una maniera non visuale
 - Per esempio utilizzando marcatori strutturali



12. Righe e bordi

```
<div class="navigation-bar">  
<hr />  
<a rel="next" href="next.html">[Next page]</a>  
<a rel="previous" href="previous.html">[Previous  
page]</a>  
<a rel="first" href="first.html">[First page]</a>  
</div>
```

- Il tag Rel indica la relazione tra il documento attuale e la risorsa specificata nell'attributo href.
- Molto utile per gli utenti che utilizzano tecnologie assistive



13. Usare i CSS + marcatura strutturale per una trasformazione elegante dei contenuti

- Grazie al posizionamento assoluto, l'ordine in cui i contenuti appaiono sullo schermo può essere differente dall'ordine in cui compaiono nel documento sorgente



13. Usare i CSS + marcatura strutturale per una trasformazione elegante dei contenuti

- Un documento dovrebbe essere leggibile anche senza CSS...



- ...è importante codificare il documento sorgente utilizzando marcatori strutturali ed ordinare i contenuti in modo logico

14. Effetti di movimento con CSS e script

- Questa sezione affronta il problema in maniera interlocutoria
- Per adesso il consiglio è quello di evitare di mostrare/nascondere dinamicamente il contenuto e cambiare la presentazione del documento (movimento e colori)



15. Fogli di stile auditivi (aural)

- Le proprietà auditive permettono di veicolare informazioni ad utenti non vedenti e a chiunque utilizzi browser vocali, quasi nello stesso modo in cui i caratteri forniscono informazioni visive



15. Fogli di stile auditivi (aural)

- La specifica **CSS2.1** considera questo tipo di media **deprecato** in favore di **speech** per il quale però non definisce proprietà!
- Il supporto degli screen reader per i fogli di stile auditivi è minimo
 - Non sono supportati da JAWS 4.51, GW Micro Window Eyes, IBM Home Page Reader e Soundlinks pwWebSpeak mentre Emacspeak,
 - Uno screen reader per Linux, li supporta



16. Rappresentazioni alternative del contenuto

- I CSS permettono di accedere a rappresentazioni alternative del contenuto specificato nei valori degli attributi, utilizzando insieme:
 - Selettori di attributo;
 - La funzione attr() e la proprietà content;
 - Le pseudo-classi :before e :after.



16. Rappresentazioni alternative del contenuto

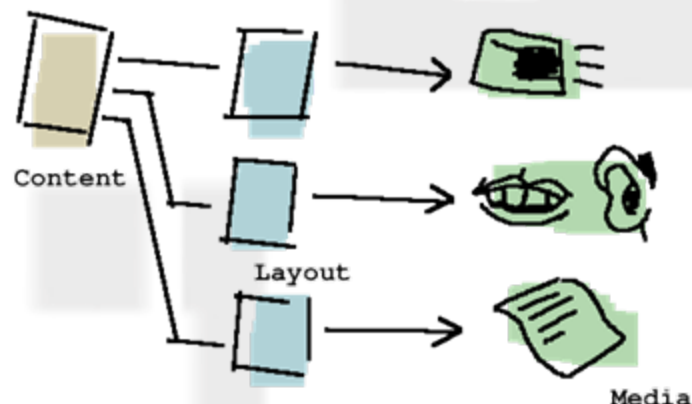
```
img:after { content: attr(alt); }
```

- Il valore dell'attributo sarà visualizzato (dopo l'immagine) anche se essa potrebbe non esserlo
 - Per esempio se l'utente ha disabilitato la visualizzazione delle immagini



17. Tipi di media

- Una caratteristica dei CSS è la possibilità di veicolare la presentazione di un documento su media differenti:
 - Schermo
 - Carta stampata
 - Sintetizzatore vocale
 - Dispositivo braille...



17. Tipi di media

- Alcune proprietà sono specifiche solo di un particolare media, altre possono essere condivise
- Per quelle condivise, servirà impostare valori differenti in funzione del media di destinazione
 - Per esempio la proprietà *font-size* è utile sia per lo schermo sia per la stampa
 - Di solito serve un carattere più largo per lo schermo che per la carta stampata



17. Tipi di media: *@media*

```
@media print { body { font-size: 10pt } }
```

```
@media screen { body { font-size: 13px } }
```

```
@media screen, print {body { line-height:  
                        1.2em;}}
```

- Questo set di regole specifica valori differenti della proprietà *font-size* di *BODY*, uno specifico per la stampa e l'altro per la visualizzazione su schermo
- Infine imposta un'interlinea identica per entrambi i media.



17. Tipi di media

- **All**
 - Adatto a qualsiasi dispositivo
- **Braille**
 - Per dispositivi braille
- **Embossed**
 - Per stampanti braille
- **Handheld**
 - Per dispositivi portatili
- **Print**
 - Per la stampa
- **Projection**
 - Per le presentazioni proiettate
- **Speech**
 - Per i sintetizzatori vocali (sostituisce il media aural della specifica CSS2)
- **Ttyper**
 - Media che utilizzano una griglia di caratteri fissa come telescriventi, terminali, o dispositivi portatili con limitate capacità di visualizzazione
- **Tv**
 - Per dispositivi di tipo televisivo (bassa risoluzione, colore, schermi con scrollabilità limitata, disponibilità del suono)



Layout: tabelle vs. CSS



*CSS e tabelle a confronto per la
creazione di un layout accessibile*

Perché si usano le tabelle di layout?

- Editor visuali
- Cattive abitudini degli sviluppatori
 - No cross-browser
 - Approccio più semplice
 - Non conoscenza dell'accessibilità



Tabelle di layout: pro

- I software WYSIWYG possono:
 - Generare layout molto complessi
 - Evitare il ritocco “manuale” del codice
- *Apparente* maggiore facilità



Tabelle di layout: contro

- **Riducono, spesso impediscono:**
 - L'organizzazione strutturata dei contenuti
 - La separazione contenuto-presentazione
- **Codice confuso e ridondante**
 - Spesso causano un aumento del peso delle pagine
- **Riducono l'accessibilità del sito**
- **Per ogni pagina è necessario ricreare il layout**
 - Cambiare la grafica di un sito richiede la revisione di **TUTTE** le pagine del sito



CSS: pro

- **Separazione fra contenuti e grafica**
- **Contenuto strutturato in sezioni e sottosezioni**
- **Codice (X)HTML semplice e pulito**
- **File più leggeri, più veloci da scaricare**
- **Diversi layout per diversi media**
- **Accesso ai contenuti con qualsiasi dispositivo**
 - Anche quelli non in grado di interpretare i CSS
- **Maggiore coerenza grafica fra le varie pagine di un sito**
- **Si cambia l'aspetto del sito modificando solamente un file**



CSS: contro

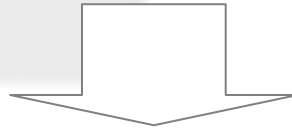
- I browser più vecchi interpretano i CSS in maniera parziale e/o errata
- Progettazione *apparentemente* meno semplice



La rivoluzione...

WYSIWYG

What you see is what you get



WYCIWYG

What you code is what you get



Layout a 3 colonne table-less

- Proviamo a realizzare un layout a tre colonne sfruttando il posizionamento assoluto
- Applicheremo tre diversi CSS per ottenere tre layout:
 - Larghezza prefissata
 - Liquido con colonne laterali di larghezza prefissata
 - Completamente liquido



```
<body>
<!-- intestazione -->
  <div id="testa"><h1>Nome del sito</h1></div>
<!-- /testa -->
  <hr />
<!-- corpo -->
  <div id="corpo">

<div id="corpo-colonna1">[...]</div>
  <hr />
<div id="corpo-colonna2">[...]</div>
  <hr />
<div id="corpo-colonna3">[...]</div>

</div>
<!-- /corpo -->
  <hr />
<!-- pie' di pagina -->
  <div id="piedipagina"><p>pié di pagina</p></div>
<!-- pie' di pagina -->
</body>
```



Layout a 3 colonne table-less

- Nei CSS innanzitutto si posiziona in maniera relativa il corpo (id="corpo")
- In questo modo, ogni posizionamento assoluto di elementi al suo interno, si riferirà al corpo stesso e non a tutta la pagina



Layout a 3 colonne table-less

- **Poi creeremo le colonne laterali, impostandone la larghezza e posizionandole in maniera assoluta**
 - A questo punto, le colonne laterali si troveranno sovrapposte alla colonna centrale
 - Per evitare tale sovrapposizione sarà sufficiente impostare i margini laterali della colonna centrale



Layout a 3 colonne table-less

- **Vantaggi:**
 - L'ordine di scrittura delle colonne non è vincolante
 - Approccio robusto nei confronti dei browser



Layout a 3 colonne table-less

- **Svantaggi:**
 - **La colonna centrale deve essere più alta di quelle laterali**
 - Altrimenti, per evitare che le colonne laterali si sovrappongano al footer, si dovrà aggiungere del padding inferiore (padding-bottom) alla colonna centrale

layout3col_struttura.htm
layout_a_tre_colonne.html



Infografia

- **www.w3.org - W3C**
- **“Accessibilità: dalla teoria alla realtà”**
(Roberto Scano – IWA Edizioni)
 - » *Cap. 13 a cura di Marco Bertoni*
- **www.constile.org**
 - » *di Gianluca Troiani*
- **www.csszengarden.com**
- **<http://www.css-discuss.org/>**
 - » *Lista moderata da Eric Meyer*
- **<http://itlists.org/mailman/listinfo/cssdesign>**
 - » *Lista in italiano*

