Tecnologie Web T
CSS

Home Page del corso: http://www-db.disi.unibo.it/courses/TW/
Versione elettronica: 1.05.CSS.pdf
Versione elettronica: 1.05.CSS-2p.pdf

A cosa servono i CSS

- I fogli di stile a cascata (Cascading Style Sheets = CSS) hanno lo scopo fondamentale di separare contenuto e presentazione nelle pagine Web
  - Il linguaggio HTML serve a definire il contenuto (! *ma anche struttura, linking e semantica*) senza tentare di dare indicazioni su come rappresentarlo
  - I CSS servono a definire come il contenuto deve essere presentato all'utente
- Il linguaggio CSS è stato proposto da *Bert Bos* e *Håkon Lie*
- La parola chiave del lingiuaggio CSS è *cascading*:
  - È prevista ed è 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

Vantaggi dei CSS

- I vantaggi della separazione di competenze sono evidenti:
  - Lo stesso contenuto diventa riusabile in più contesti
  - Basta cambiare i CSS e può essere presentato correttamente in modo ottimale su dispositivi diversi (es. PC e palmari) o addirittura su media diversi (es. video e carta)
  - Si può dividere il lavoro fra chi gestisce il contenuto e chi si occupa della parte grafica

Altri obiettivi dei CSS

- Tra gli altri obiettivi:
  - Ridurre i tempi di scaricamento delle pagine: una pagina che usa i CSS è meno della metà di una pagina che usa la formattazione con tag HTML, inoltre se il file CSS è condiviso da più pagine viene scaricato una volta sola
  - Ripulire il codice HTML: eliminare l'uso di estensioni «non proprietarie»
  - Rendere le pagine «visualizzabili» e «usabili» da dispositivi non convenzionali: laptop, palmari, smartphone, ecc.

CSS: caratteristiche

- Tra le caratteristiche principali dei CSS:
  - Il controllo sia dell'autore sia del lettore di un documento HTML
  - Il fatto di essere indipendenti 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

Un po' di storia…

- Le specifiche del CSS sono state emanate dal W3C
  - 1996 - CSS level 1 (W3C Rec.) 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
  - 1998 - CSS level 2 (W3C Rec.), invece *introduce il supporto per media multipli e un linguaggio di layout sofisticato e complesso*
  - 2004 - CSS level 2.1 (W3C Rec.) versione migliorativa del CSS level 2
    - diviene W3C Rec. solo nel 2011

Un po' di storia: continua…

- 2009 - CSS level 3: è stato profondamente coinvolto nel processo di standardizzazione di HTML 5, è quindi fortemente interconnesso con lo sviluppo del linguaggio
  - CSS level 3 estende le funzionalità CSS 2.1 mantenendone la piena compatibilità
- A differenza di CSS 2.1, che si presentata come unica grande specifica, CSS level 3 è diviso in diversi moduli (dei quali 4 sono W3C Recommendation – a partire dal 2011)

- Nel seguito faremo riferimento alle specifiche CSS 2.1 per poi introdurre CSS 3 per differenze rispetto a CSS 2.1
  - Analogamente a quanto già fatto per HTML 4.01 e HTML 5
- Analizzeremo poi esempi rilevanti di applicazione di HTML 5 + CSS level 3 durante le esercitazioni guidate in laboratorio…

! *CSS e compatibilità browser*

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

- Alle pagine:
  `http://caniuse.com`
  `http://www.w3schools.com/css/`
  `http://www.w3schools.com/css3/`

- è possibile trovare una versione ragionevolmente affidabile del supporto dei browser alle singole feature delle varie versioni di CSS e vari esempi di utilizzo

CSS e HTML

```
<html>
  <head>...</head>
  <body>
    <h1>title</h1>
    <div>
      <p> uno </p>
      <p> due </p>
    </div>
    <p> tre
      <a href="link.html">link</a>
    </p>
  </body>
</html>
```

Un documento HTML può essere visto come un insieme di blocchi (contenitori) sui quali si può agire con stili diversi.

Ogni tag HTML definisce un blocco.

Un esempio: Hello World

- Creiamo una pagina HTML - denominata hello.html che mostra la scritta *Hello World! Usiamo i CSS*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
     <TITLE>Hello World</TITLE>
  </HEAD>
  <BODY>
     <H1>Hello World!</H1>
     <p>Usiamo i CSS</p>
  </BODY>
</HTML>
```

- Non abbiamo dato alcuna indicazione su come rappresentare la pagina
- Abbiamo solo specificato che si tratta di un titolo di primo livello e di un paragrafo
- Il browser userà gli stili standard

Hello World con CSS

- Creiamo un secondo file di testo – denominato hello.css - che contiene queste due righe:

```
BODY { color: red }
H1 { color: blue }
```

- Colleghiamo i due file inserendo il link al CSS nella testata della pagina hello.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Hello World con CSS</TITLE>
    <LINK rel="stylesheet" href="hello.css" type="text/css">
  </HEAD>
  <BODY>
    <H1>Hello World!</H1>
    <p>Usiamo i CSS</p>
  </BODY>
</HTML>
```

Il risultato

- Ecco il risultato nei due casi:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
     <TITLE>Hello World</TITLE>
  </HEAD>
  <BODY>
     <H1>Hello World!</H1>
     <p>Usiamo i CSS</p>
  </BODY>
</HTML>
```

hello.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Hello World con CSS</TITLE>
    <LINK rel="stylesheet" href="hello.css" type="text/css">
  </HEAD>
  <BODY>
    <H1>Hello World!</H1>
    <p>Usiamo i CSS</p>
  </BODY>
</HTML>
```

hello.html

```
BODY { color: red }
H1 { color: blue }
```

hello.css

Applicare gli stili ad una pagina - 1

- Abbiamo due possibilità:
  - Mettere gli stili in uno o più file separati
  - Inserire gli stili nella pagina stessa
- Se si sceglie di mettere gli stili in file separati (si parla di stili esterni) sono possibili due sintassi diverse:

```
<HTML>
  <HEAD>
    <TITLE>Hello World</TITLE>
    <link rel="stylesheet"
      href="hello.css"
      type="text/css">
  </HEAD>
  <BODY>
    <H1>Hello World!</H1>
    <p>Usiamo i CSS</p>
  </BODY>
</HTML>
```

```
<HTML>
  <HEAD>
    <TITLE>Hello World</TITLE>
    <style type="text/css">
      @import url(hello.css);
    </style>
  </HEAD>
  <BODY>
    <H1>Hello World!</H1>
    <p>Usiamo i CSS</p>
  </BODY>
</HTML>
```

Applicare gli stili ad una pagina - 2

- Se invece si sceglie di mettere gli stili nella pagina si può procedere in due modi:
  - Mettere tutti gli stili nell'header in un tag <style> (stili interni)
  - Inserirli nei singoli elementi (stili inline)

```
<HTML>
  <HEAD>
    <TITLE>Hello World</TITLE>
    <style type="text/css">
      BODY { color: red }
      H1 { color: blue }
    </style>
  </HEAD>
  <BODY>
    <H1>Hello World!</H1>
    <p>Usiamo i CSS</p>
  </BODY>
</HTML>
```

```
<HTML>
  <HEAD>
    <TITLE>Hello World</TITLE>
  </HEAD>
  <BODY style="color: red">
    <H1 style="color: blue">
       Hello World!</H1>
    <p>Usiamo i CSS</p>
  </BODY>
</HTML>
```

Qual è la scelta migliore?

- È sicuramente preferibile usare gli stili esterni:
  - È facile applicare diversi stili alla stessa pagina
  - Si ottimizza il trasferimento delle pagine perché il foglio stile rimane nella cache del browser
- L'uso degli stili inline è da evitare
  - rendono molto basso il livello di separazione fra contenuto e presentazione
  - Le modifiche sono molto complicate
- Gli stili interni sono una via di mezzo
- Fra le due sintassi per gli stili esterni:
  - Quella con <link> è più diffusa
  - Quella con @import è meno critica per la compatibilità con i vecchi browser

Regole e loro struttura

- Un'espressione come **`H1 { color: blue }`** prende il nome di regola CSS
- Una regola CSS è composta da due parti:
  - Selettore: **`H1`**
  - Dichiarazione: **`color: blue`**
- La dichiarazione a sua volta si divide in due parti
  - Proprietà: **`color`**
  - Valore: **`blue`**
- La sintassi generale si può quindi esprimere così

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

- O più in generale:

```
selettore {
    proprietà1 : valore1;
    proprietà2 : valore2, valore3; }
```

Selettori - 1

- Il selettore serve per collegare uno stile agli elementi a cui deve essere applicato
- Selettore universale: identifica qualunque elemento

```
* { … }
```

- Selettori di tipo: si applicano a tutti gli elementi di un determinato tipo (ad es. tutti i <p>)

```
tipo_elemento { … }
```

- Classi: si applicano a tutti gli elementi che presentano l'attributo `class="nome_class"`

```
.nome_classe { … }
```

- Identificatori: si applicano agli elementi che presentano l'attributo `id="nome_id"`

```
#nome_id { … }
```

Combinazioni di selettori

- I selettori di tipo si possono combinare con quelli di classe e di identificatore:

```
tipo_elemento.nome_classe { … }
tipo_elemento#nome_id { … }
```

- In effetti negli esempi della pagina precedente la mancanza di un tipo_elemento prima di . / # sottintendeva la presenza del selettore universale *

```
.nome_classe   →   *.nome_classe
#nome_id       →   *.#nome_id
```

Selettori - 2

- Pseudoclassi: si applicano ad un sottoinsieme degli elementi di un tipo identificato da una proprietà

```
tipo_elemento:proprietà { … }
```

  - Es. stato di un'ancora: link e visited

```
a:link { … }, a:visited { … }
```

  - Es. condizione di un elemento: active, focus e hover

```
h1:hover { … },
```

- Pseudoelementi: si applicano ad una parte di un elemento

```
tipo_elemento:parte { … }
```

  - Es. solo la prima linea o la prima lettera di un paragrafo:

```
p:first-line { … }
p:first-letter { … }
```

Selettori - 3

- Selettori gerarchici: si applicano a tutti gli elementi di un dato tipo che hanno un determinato legame gerarchico (discendente, figlio, fratello) con elementi di un altro tipo
  - **`tipo1 tipo2 { … }`** tipo2 discende da tipo1
  - **`tipo1>tipo2 { … }`** tipo2 è figlio di tipo1
  - **`tipo1+tipo2 { … }`** tipo2 è fratello di tipo1
- Ad esempio: **`UL>LI { … }`** si applica solo agli elementi contenuti direttamente in liste non ordinate:

```
<UL>
  <LI>Riga 1</LI>
</UL>
```

SI

```
<UL>
  <OL>
      <LI>Riga 1</LI>
  </OL>
</UL>
```

NO

Raggruppamenti

- Se la stessa dichiarazione si applica a più tipi di elemento si scrive una regola in forma raggruppata

```
H1 { font-family: sans-serif }
H2 { font-family: sans-serif }
H3 { font-family: sans-serif }
```

equivale a

```
H1, H2, H3 { font-family: sans-serif }
```

Proprietà

- Nelle dichiarazioni è possibile far uso di proprietà singole o in forma abbreviata (shorthand properties)
  - Le proprietà singole permettono di definire un singolo aspetto di stile
  - Le shorthand properties consentono invece di definire un insieme di aspetti, correlati fra di loro usando una sola proprietà
- Per esempio, ogni elemento permette di definire un margine rispetto a quelli adiacenti usando quattro proprietà:
  `margin-top, margin-right, margin-bottom, margin-left`
- Utilizziamo le proprietà singole applicandole ad un paragrafo:

```
P { margin-top: 10px;
    margin-right: 8px;
    margin-bottom: 10px;
    margin-left: 8px; }
```

- Lo stesso risultato si può ottenere usando la proprietà in forma abbreviata **`margin`**:

```
P {margin: 10px 8px 10px 8px;}
```

Valori - 1

- Numeri interi e reali: “.” come separatore decimale
- Grandezze: usate per lunghezze orizzontali e verticali
  - un numero seguito da una unità di misura
- Unità di misura relative
  - em: è relativa alla dimensione del font in uso
    (es. se il font ha corpo 12pt, 1em varrà 12pt, 2em varranno 24pt)
  - px: pixel, sono relativi al dispositivo di output e alle impostazioni del computer dell'utente
- Unità di misura assolute
  - in: pollici; (1 in =2.54 cm)
  - cm: centimetri
  - mm: millimetri
  - pt: punti tipografici (1/72 di pollice)
  - pc: pica = 12 punti

Valori - 2

- Percentuali: percentuale del valore che assume la proprietà stessa nell'elemento padre; un numero seguito da %
- URL assoluti o relativi; si usa la notazione **`url(percorso)`**
- Stringhe: testo delimitato da apici singoli o doppi
- Colori: possono essere identificati con tre metodi differenti:
  - In forma esadecimale **`#RRGGBB`**
  - Tramite la funzione **`rgb(rosso,verde,blu)`**
  - Usando una serie di parole chiave che possono indicare colori assoluti o dipendenti dall'impostazione del PC (proprietà di sistema)

Colori: parole chiave

- **Colori assoluti:**

black - nero	green - verde
silver - argento	lime - verde chiaro
gray - grigio	olive - oliva
white - bianco	yellow - giallo
maroon - marrone	navy - blu scuro
red - rosso	blue - blu
purple - viola	teal - verde acqua scuro
fuchsia - fucsia	aqua - verde acqua

- **Colori dipendenti dalle proprietà di sistema:**

background - il colore di sfondo del desktop
buttonFace - il colore di sfondo dei pulsanti
buttonText - testo dei pulsanti
captionText - testo delle etichette
grayText - testo disabilitato

Attribuzione di uno stile ad un elemento

- Per poter rappresentare una pagina HTML il browser deve riuscire ad applicare ad ogni elemento uno stile
- Un elemento privo di stile non può essere rappresentato:
  - Anche se nella pagina non c'è nessuna regola CSS (interna o esterna) ogni browser ha un foglio stile di default che contiene stili per ogni tipologia di elemento HTML (tag)
- L'attribuzione può essere diretta:
  - L'elemento contiene uno stile inline
  - Esistono una o più regole il cui selettore rimanda all'elemento
- Oppure può essere indiretta:
  - L'elemento "eredita" lo stile dall'elemento che lo contiene

Ereditarietà

- È un meccanismo di tipo differenziale simile per certi aspetti all'ereditarietà nei linguaggi ad oggetti
- Si basa sui blocchi annidati di un documento HTML
  - Uno stile applicato ad un blocco esterno si applica anche ai blocchi in esso contenuti
- In un blocco interno:
  - Si possono definire stili aggiuntivi
  - Si possono ridefinire stili definiti in un blocco più esterno (è una sorta di *overriding*)
- Lo stesso ragionamento si può esprimere in termini di DOM:
  - un nodo figlio eredita gli stili dei nodi che si trovano sul ramo da cui discende

Esempio di ereditarietà

- Torniamo all'esempio visto all'inizio…

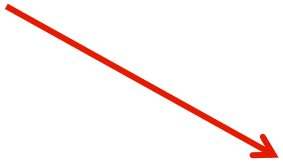
- L'elemento Usiamo i CSS non ridefinisce il colore del testo e quindi eredita da <body>: viene mostrato in rosso
- L'elemento <H1>Hello World</H1> ridefinisce lo stile e quindi appare in blu

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Hello World con CSS</TITLE>
    <LINK rel="stylesheet"
     href="hello.css" type="text/css">
  </HEAD>
  <BODY>
    <H1>Hello World!</H1>
    <p>Usiamo i CSS</p>
  </BODY>
</HTML>
```

hello.html

```
BODY { color: red }
H1 { color: blue }
```

hello.css



Hello World!

Usiamo i CSS

Limitazioni all'ereditarietà

- Non tutte le proprietà sono soggette al meccanismo dell'ereditarietà
- In generale non vengono ereditate quelle che riguardano la formattazione del box model
- Il box è il riquadro che circonda ogni elemento
- La motivazione è abbastanza intuitiva: se ogni elemento interno ereditasse le proprietà dell'elemento che lo contiene avremmo un effetto grafico tipo "scatole cinesi" assolutamente indesiderato

Conflitti di stile

- Nell'applicare gli stili possono nascere conflitti di competenza per diversi motivi:
  - Esiste un'intersezione tra regole che utilizzano selettori di tipo diverso, ad esempio ID e Classe come in questo caso:

```
p#myID {text-color=red}
p.myClass {text-color=blue}
…
<p id=myID class=myClass>
```

  - Una pagina usa più fogli di stile oppure combina fogli di stile esterni e regole interne o inline
  - Nello stesso foglio stile ci sono regole con lo stesso selettore e dichiarazioni diverse
    - banale errore o gestione disordinata dei CSS

Cascade

- Lo standard CSS definisce un insieme di regole di risoluzione dei conflitti che prende il nome di cascade
- La logica di risoluzione si basa su tre elementi
- Origine del foglio stile:
  - Autore: stile definito nella pagina
  - Browser: foglio stile predefinito
  - Utente: in alcuni browser si può editare lo stile
- Specificità: esiste una formula che misura il grado di specificità attribuendo, ad es., un punteggio maggiore ad un selettore di ID rispetto ad uno di Classe
- Dichiarazione !important: è possibile aggiungere al valore di una dichiarazione la clausola !important

```
p.myClass {text-color: red !important}
```

Regole di risoluzione dei conflitti

- Il CSS assegna un peso a ciascun blocco di regole
- In caso di conflitto vince quella con peso maggiore
- Per determinare il peso si applicano in sequenza una serie di regole:
  - Origine: l'ordine di prevalenza è *autore, utente, browser*
  - Specificità del selettore: ha la precedenza il selettore con specificità maggiore
  - Ordine di dichiarazione: se esistono due dichiarazioni con ugual specificità e origine vince quella fornita per ultima.

  - *N.B. Le dichiarazioni esterne sono considerate come precedenti a qualsiasi dichiarazione interna*

Effetto di !important

- L’effetto della clausola !important è molto semplice
  - Una regola marcata come !important ha sempre precedenza sulle altre, indipendentemente da origine, specificità e ordine di apparizione

Proprietà

- CSS definisce una sessantina di proprietà che ricadono grosso modo nei seguenti gruppi:
  - Colori e sfondi
  - Caratteri e testo
  - Box model
  - Liste
  - Display e gestione degli elementi floating
  - Posizionamento
  - Tabelle

Color

- Per ogni elemento si possono definire almeno tre colori:
  - il colore di primo piano (foreground color)
  - il colore di sfondo (background color)
  - il colore del bordo (border color)
- La proprietà color definisce esclusivamente:
  - il colore di primo piano, ovvero quello del testo
  - il colore del bordo di un elemento quando non viene impostato esplicitamente con border-color
- La sintassi di color è:

```
selettore { color: <valore> }
```

dove il valore è definito con le modalità descritte in precedenza (parola chiave, #RRGGBB, ecc.)

Background

- La definizione dello sfondo può essere applicata a due soli elementi: body e tabelle
- Proprietà singole e valori:
  - **`background-color`**: colore oppure **`transparent`**
  - **`background-image`**: url di un'immagine o **`none`**
  - **`background-repeat`**:**`{repeat|repeat-x|repeat-y|no-repeat}`**
  - **`background-attachment`**: **`{scroll|fixed}`**
  - **`background-position`**: **`x,y`** in % o assoluti o parole chiave **`(top|left|bottom|right}`**
- Proprietà in forma breve: **`background`**

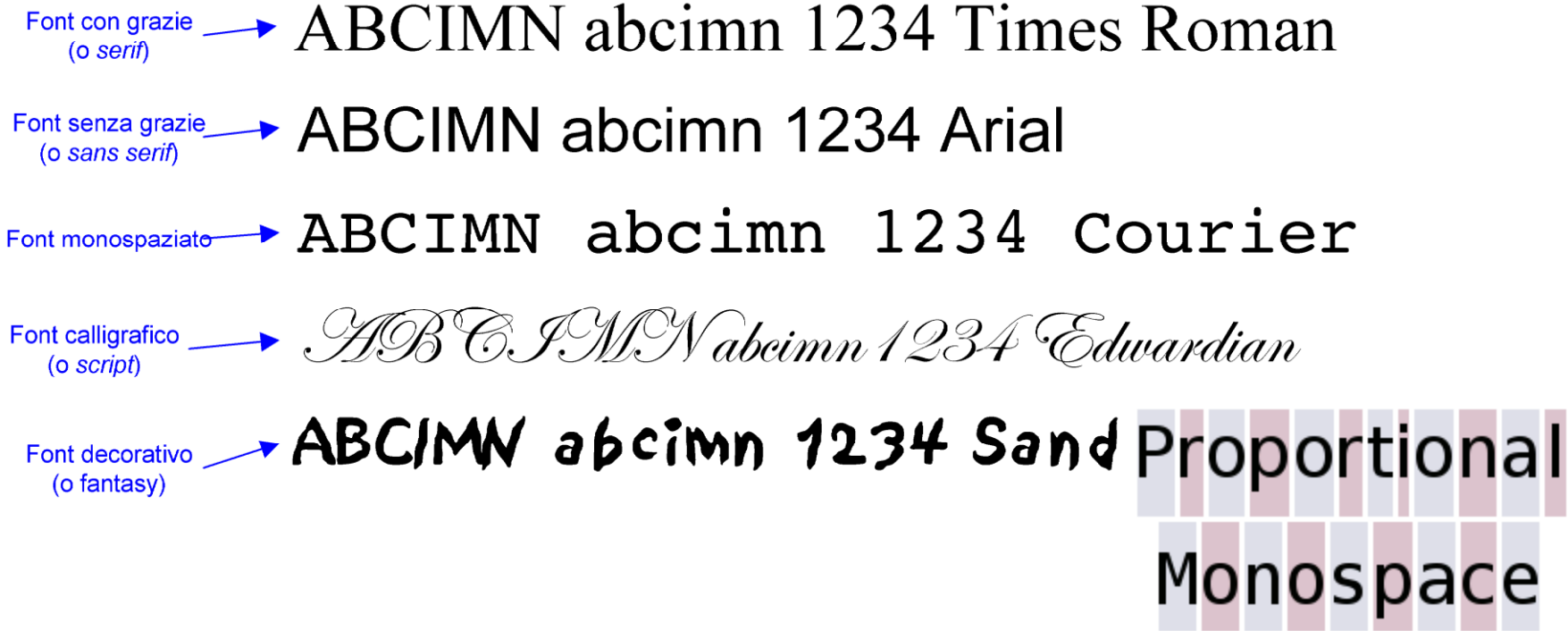
```
selettore {background: background-color background-image background-repeat background-attachment background-position;}
```

Gestione del testo

- Una parte consistente di CSS tratta la gestione del testo
- Esistono proprietà per definire tutti gli elementi classici della tipografia
- Aspetto dei caratteri:
  - Tipo di carattere (font)
  - Dimensione
  - Peso
  - Varianti di stile (normale, corsivo)
- Formattazione del testo:
  - Interlinea
  - Allineamento
  - Rientri
  - Decorazioni (sottolineato, barrato, ecc.)

Font

- Un font è una serie completa di caratteri (lettere, cifre, segni di interpunzione) con lo stesso stile
- Possono essere classificati sulla base di diversi criteri
  - presenza o assenza di grazie o “bastone” (inglese *sans serif* (talvolta solo *sans*)
  - spaziatura fissa o proporzionale



Con grazie o senza grazie?

- Le grazie sono piccole decorazioni che sporgono dal corpo della lettera (in rosso nella figura a lato)
- Nei testi stampati ad alta risoluzione i caratteri con grazie risultano molto più leggibili (le grazie “guidano” l’occhio)
- Quando però si lavora a bassa risoluzione (i video dei computer sono tutti a bassa risoluzione) i caratteri senza grazie risultano molto più leggibili
- Esistono alcuni font senza grazie, per esempio il Verdana, che sono stati pensati per essere molto leggibili anche a video e con caratteri di piccola dimensione

font-family - 1

- La proprietà che ci permette di definire il tipo di carattere è **`font-family`** che prende come valore il nome di un font:

```
p {font-family: Verdana}
```

- I font pongono un problema di compatibilità piuttosto complesso
  - su piattaforme diverse (Windows, Mac, Linux…) sono disponibili caratteri diversi e ogni utente può avere un proprio set personalizzato
- Per gestire questa situazione CSS mette a disposizione due meccanismi
  - La definizione di famiglie generiche
  - La possibilità di dichiarare più font in una proprietà

font-family - 2

- Le 5 famiglie generiche sono e hanno una corrispondenza specifica che dipende dalla piattaforma (fra parentesi i valori utilizzati da Windows):
  - serif (Times New Roman)
  - sans-serif (Arial)
  - cursive (Comic Sans)
  - fantasy (Allegro BT)
  - monospace (Courier)
- Una dichiarazione multipla è fatta in questo modo:

```
p {font-family: Verdana, Helvetica, sans-serif;}
```

- Il browser procede per ordine: cerca prima il font Verdana, altrimenti cerca Helvetica e se manca anche questo ricorre all'ultimo tipo sans-serif
- sans-serif è una famiglia generica e quindi si trova sempre una corrispondenza
- Conviene quindi mettere sempre per ultima una famiglia generica

font-size

- La proprietà **`font-size`** permette di definire le dimensioni del testo (in tipografia: corpo del carattere)
- La dimensione può essere espressa in forma assoluta:
  - con una serie di parole chiave
    - **`xx-small, x-small, small, medium, large, x-large, xx-large`**
  - con unità di misura assolute
    - tipicamente pixel (**`px`**) e punti (**`pt`**)
- Oppure in forma relativa:
  - con le parole chiave **`smaller`** e **`larger`**
  - con l'unità **`em`**: proporzione rispetto al valore del font corrente (basato sulla M maiuscola)
    - es. **`1.5em`** = una volta e mezzo
  - con l'unità **`ex`**: proporzione rispetto all'altezza del corpo della x minuscola del font corrente
  - in percentuale (% rispetto al valore corrente: **`75%`**)

Il corpo

- Il corpo del carattere tiene conto di una serie di fattori:
  - Ascendenti (puntino della i o asta della d)
  - Discendenti (gamba della g o della q)
  - Accenti delle maiuscole

font-size e compatibilità

- Qual è il miglior modo di definire il corpo di un carattere?
- In teoria sullo schermo sarebbe bene usare i pixel, ma IE non consente all'utente di ridimensionare un testo espresso in pixel
- La scelta migliore (consigliata da W3C) è quella di utilizzare l'em
  - Esprimere in % la dimensione del testo nel body (tipicamente 100%) e poi usare gli em per gli elementi interni

```
body {font-size:100%}
h1   {font-size:2.5em}
p    {font-size:0.875em}
```

font-weight

- La proprietà **`font-weight`** definisce il peso del carattere (la grossezza dei tratti che lo compongono)
- L'esempio più noto è normale/neretto (impropriamente chiamato grassetto) ma è possibile avere una gamma più ampia di pesi
- Il peso si può esprimere in diversi modi:
  - Valori numerici: **`100, 200, … 800, 900`**
  - Parole chiave assolute: **`normal, bo`**
  - Parole chiave relative: **`bolder, lighter`**
- **`normal`** corrisponde a 400, **`bold`** a 700

Avenir Next Ultralight
Avenir Next Regular
Avenir Next Medium
Avenir Next Demi
Avenir Next Bold
Avenir Next Heavy

100 · why pangolins dream of quiche
200 · why pangolins dream of quiche
300 · why pangolins dream of quiche
400 · why pangolins dream of quiche
500 · why pangolins dream of quiche
600 · why pangolins dream of quiche
700 · why pangolins dream of quiche
800 · why pangolins dream of quiche
900 · why pangolins dream of quiche

font-style

- La proprietà **`font-style`** permette di definire varianti del testo rispetto al normale (tondo nel linguaggio tipografico)
  - normal: valore di default (***tondo***)
  - italic: testo in corsivo
  - oblique: testo obliquo, simile a italic
- **In tipografia:**
  - il corsivo viene progettato separatamente e può essere anche molto diverso dal tondo:

Prova ***Prova***

  - Il testo obliquo è invece derivato per deformazione dal tondo

The five boxing

The five boxing

The five boxing

font-variant

- La proprietà **`font-variant`** è simile a **`font-style`** e permette di impostare un'altra variante del testo: il maiuscoletto (small-caps)
- Ammette due valori: **`normal`** e **`small-caps`**
- Come per il corsivo anche per il maiuscoletto in tipografia abbiamo una variante progettata ad-hoc (in pratica per un font si ha la serie dei caratteri maiuscoli, quella dei minuscoli e quella dei maiuscoletti)
- Nei computer però questa variante non è quasi mai disponibile e si usa un carattere maiuscolo di un corpo più piccolo (falso maiuscoletto)

La proprietà font

- La proprietà **`font`** è una proprietà sintetica che consente di definire tutti gli attributi dei caratteri in un colpo solo, nell'ordine:
  - **`font-style font-variant font-weight font-size font-family font di sistema`**
- Es: `p {font: italic normal bold 10px Arial,Serif;}`
- I font di sistema permettono di adattare le pagine all'aspetto del sistema operativo, sono 6 valori:
  - caption: font usato per bottoni e combo-box
  - icon: font usato per il testo delle icone
  - menu: carattere usato nei menu delle varie finestre
  - message-box: carattere usato per i popup
  - small-caption: carattere per i controlli più piccoli
  - status-bar: font usato per la barra di stato

line-height

- **`line-height`** permette di definire l'altezza di una riga di testo all'interno di un elemento blocco
- In pratica consente di definire l'interlinea (lo spazio fra le righe) usando i seguenti valori:
  - normal: spaziatura di default stabilita dal browser
  - valore numerico: moltiplicatore applicato al corpo del carattere (es. 1.5= una volta e mezza il corpo)
  - valore con unità di misura: altezza esatta della riga
  - percentuale: altezza riga pari a una % del corpo
- **In tipografia la spaziatura *giusta* è uguale al corpo, un testo con spaziatura maggiore si dice *interlineato*, con spaziatura minore si dice *sterlineato***
- La scelta migliore è il valore numerico

```
p {line-height: 1.5;}
p {line-height: 15px;}
```

Allineamento e decorazione del testo

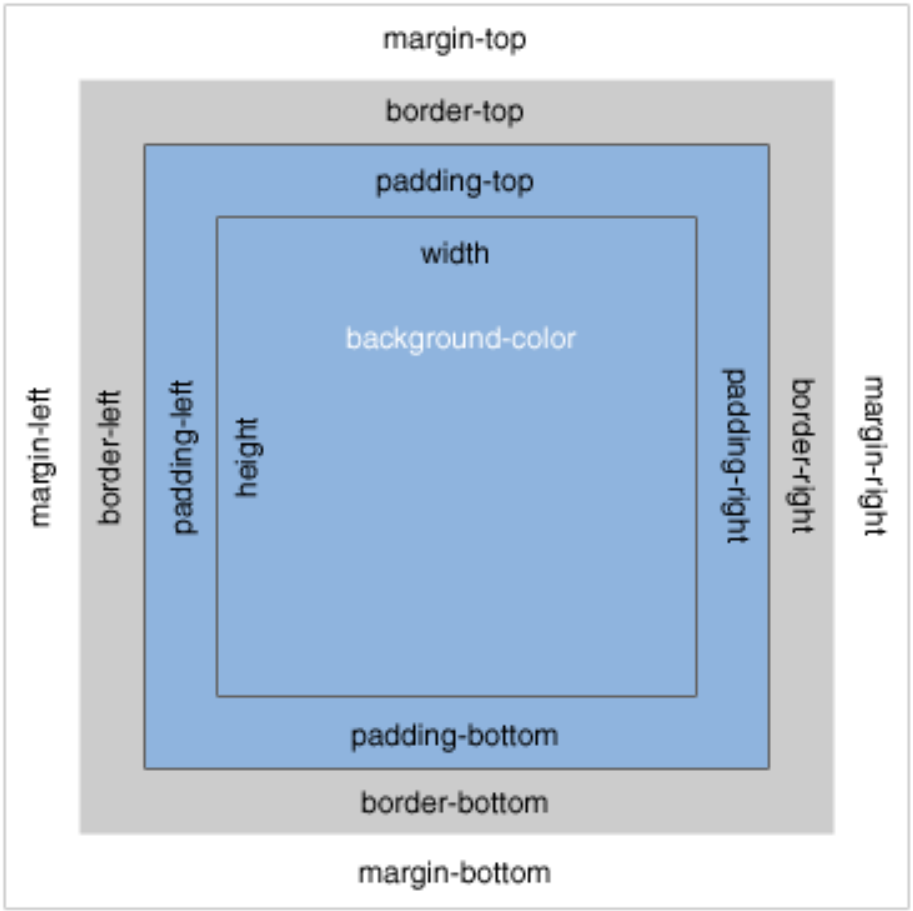
- Con **`text-align`** possiamo definire l'allineamento di un paragrafo scegliendo fra 4 opzioni:
  - left: allineamento a sinistra (***bandiera a sinistra***)
  - right: allineamento a destra (***bandiera a destra***)
  - center: centratura (***epigrafe***)
  - justify: giustificazione (***blocchetto***)
- **`text-decoration`** permette invece di definire alcune decorazioni (sottolineato, barrato ecc.):
  - none: nessuna decorazione
  - underline: sottolineato
  - overline: linea sopra il testo
  - line-through: barrato

text-indent e text-transform

- **`text-indent`** definisce l'indentazione della prima riga in ogni elemento contenente del testo
- Può essere espressa in due modi
  - valore numerico con unità di misura
  - valore in percentuale rispetto alla larghezza del blocco di testo (***giustezza***)
- **`text-transform`** serve a cambiare gli attributi del testo relativamente a maiuscole e minuscole
  - none: nessuna trasformazione
  - capitalize: la prima lettera di ogni parola in maiuscolo, le altre tutte in minuscolo
  - uppercase: tutto il testo è maiuscolo
  - lowercase: tutto il testo è minuscolo
- Utile, ad esempio, per dare un aspetto accettabile a pezzi di testo tutti in maiuscolo

Il modello dei riquadri (box model)

- Per box model si intende l'insieme delle regole per la definizione degli stili per gli elementi blocco
- Il modello comprende cinque elementi base rappresentati nella figura



Box model - Elementi

- Area del contenuto: testo, immagine, ecc. di cui è possibile definire altezza e larghezza (width e height)
- Padding (cuscinetto): spazio vuoto tra il contenuto e il bordo dell'elemento
- Bordo (border): di cui possiamo definire colore, stile e spessore
- Margine (margin): spazio tra l'elemento e gli altri elementi adiacenti
  - Nel caso di due box allineati in orizzontale, la loro distanza è la somma dei due margini
  - Se sono allineati verticalmente, si ha il cosiddetto margin collapsing
    - la distanza è pari al massimo fra il margine inferiore del primo e il margine superiore del secondo

Larghezza e altezza del box

- La larghezza complessiva è data dalla formula:

```
margine sx + bordo sx + padding sx +
width +
padding dx + bordo dx + margine dx
```

- Se non si imposta specificamente il valore width (o si specifica il valore auto) la dimensione del contenuto viene stabilita automaticamente dal browser
- Per l'altezza complessiva vale un ragionamento analogo ma bisogna tener conto del margin collapsing per cui il valore dipende anche da cosa c'è vicino

Gestione del box model: dimensioni del contenuto

- **`height`**: altezza, si applica a tutti gli elementi blocco escluse le colonne delle tabelle
- **`min-height`** e **`max-height`**: permettono di fissare l'altezza minima o quella massima anziché un valore esatto (**`min-height`** non funziona con IE)
- **`width`**: larghezza
- **`min-width`** e **`max-width`**: permettono di fissare la larghezza minima o quella max anziché quella esatta
- Valori ammessi:
  - auto: dimensione determinata dal contenuto (solo per **`width`** e **`height`**)
  - valore numerico con unità di misura
  - valore percentuale

! *Le proprietà del box model non vengono ereditate*

Overflow

- La proprietà **`overflow`** permette di definire il comportamento da adottare quando il contenuto (tipicamente testo) deborda dalle dimensioni fissate
- Valori:
  - visible (default): il contenuto eccedente viene mostrato, i browser si comportano in modi diversi!
  - hidden: il contenuto eccedente non viene mostrato
  - scroll: vengono mostrate barre di scorrimento che consentono di accedere al contenuto eccedente
  - auto: Il browser tratta il contenuto eccedente secondo le sue impostazioni predefinite (di solito barre di scorrimento)

! *In condizioni normali non conviene definire l'altezza in modo fisso… si creano effetti imprevedibili*

Margini

- I margini consentono di definire la spaziatura fra elementi
- Quattro proprietà singole: **`margin-top, margin-right, margin-bottom, margin-left`**
- Una proprietà sintetica: **`margin`** (con i valori nell'ordine esposto sopra)
- Valori
  - valore numerico con unità di misura
  - valore in percentuale
  - auto: distanza automaticamente calcolata rispetto alla larghezza dell'elemento contenitore

```
div {
margin-top: 8px;
margin-right: 16px;
margin-bottom: 8px;
margin-left: 24px; }
```

Padding

- Il padding consente di definire lo spazio intorno ad un elemento (internamente al bordo)
- Quattro proprietà singole: **`padding-top, padding-right, padding-bottom, margin-left`**
- Una proprietà sintetica: **`padding`** (**con** i valori nell'ordine esposto sopra)
- Valori
  - valore numerico con unità di misura
  - valore in percentuale.
  - auto: distanza automaticamente calcolata rispetto alla larghezza dell'elemento contenitore

```
div {
padding-top: 8px;
padding-right: 16px;
padding-bottom: 8px;
padding-left: 24px; }
```

Bordi

- Le proprietà di bordo permettono di definire: stile, colore e spessore di ognuno dei quattro bordi
- Dodici proprietà singole (3 per ogni bordo):
  - **`border-top-color, border-top-style, border-top-width`**
  - **`border-right-color, border-right-style, border-right-width`**
  - …
- Proprietà sintetiche di tre tipi:
  - **`border-top, border-right, border-bottom, border-left`**
  - **`border-color, border-width, border-style`**
  - **`border`**: si può usare solo quando i 4 bordi hanno le stesse caratteristiche

Valori per i bordi

- Colore:
  - colore (espresso nei vari modi possibili)
- Stile:
  - none o hidden: nessun bordo e spessore pari a 0
  - dotted, dashed: a puntini, a trattini
  - solid: intero
  - double: doppio
  - groove, ridge, inset, outset: effetti tridimensionali
- Spessore:
  - valore numerico con unità di misura
  - thin: bordo sottile
  - medium: bordo di medio spessore
  - thick: bordo molto spesso

Esempi di stili per i bordi

```
<style type="text/css">
  p.solid  { border: solid thin red; padding: 4 px; }
  p.dotted { border: dotted thin red; padding: 4 px; }
  p.dashed { border: dashed thin red; padding: 4 px; }
  p.double { border: double thick red; padding: 4 px; }
  p.groove { border: groove thick red; padding: 4 px; }
  p.ridge  { border: ridge thick red; padding: 4 px; }
  p.inset  { border: inset thick red; padding: 4 px; }
  p.outset { border: outset thick red; padding: 4 px; }
</style>
```

el Patrimonio Industriale propone su prenotazione
dicato ai bambini dai 7 ai 10 anni.

el Patrimonio Industriale propone su prenotazione
dicato ai bambini dai 7 ai 10 anni.

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione (massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni.

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione (massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni.

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione (massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni.

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione (massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni.

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione (massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni.

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione (massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni.

Liste - 1

- CSS definisce alcune proprietà che agiscono sulle liste puntate (<ul>) e numerate (<ol>), o meglio sugli elementi delle liste (<li>)
- In virtù dell'ereditarietà, se applichiamo una proprietà alle liste la applichiamo a tutti gli elementi
- **`list-style-image`**: definisce l'immagine da utilizzare come "punto elenco" e ammette i valori:
  - **`url(<url_immagine>)`**
  - **`none`**
- **`list-style-position`**: indica la posizione del punto e ammette i valori
  - **`inside`**: il punto fa parte del testo
  - **`outside`**: il punto è esterno al testo

Liste - 2

- **`list-style-type`**: aspetto del punto-elenco
- I valori possibili sono molti, ne citiamo alcuni:
  - none: nessun punto
  - disc, circle, square: cerchietto pieno, cerchietto vuoto, quadratino
  - decimal: conteggio con cifre arabe 1, 2, 3…
  - decimal-leading-zero: cifre arabe precedute da zero: 01, 02…
  - lower-roman: cifre romane in minuscolo i, ii, iii…
  - upper-roman: cifre romane in maiuscolo I, II, III...
  - lower-alpha, lower-latin: lettere minuscole a, b...
  - upper-alpha, upper-latin: lettere maiuscole A, B...
  - lower-greek: lettere minuscole in greco antico

! *Il colore può essere modificato per tutti i tipi con la proprietà* ***`color`***

display

- HTML classifica gli elementi in tre categorie: blocco, inline e lista
- Ogni elemento appartiene per default ad una di queste categorie ma la proprietà **`display`** permette di cambiare questa appartenenza
- I valori più comuni sono:
  - inline: l'elemento diventa di tipo inline
  - block: l'elemento diventa di tipo blocco
  - list-item: l'elemento diventa di tipo lista
  - none: l'elemento viene trattato come se non ci fosse (non viene mostrato e non genera alcun box)

! *Attenzione: dire che l'elemento è trattato come non presente è diverso da dire che è nascosto*

float

- Con **`float`** è possibile estrarre un elemento dal normale flusso del documento e spostarlo su uno dei lati (destro o sinistro) del suo elemento contenitore
- Il contenuto che circonda l'elemento scorre intorno ad esso sul lato opposto rispetto a quello indicato come valore di float.
  La proprietà non è ereditata
- In HTML questa possibilità era riservate alle immagini marcate con l'attributo align: CSS lo estende a tutti gli elementi
- Valori
  - left: l'elemento viene spostato sul lato sinistro del box contenitore, il contenuto scorre a destra
  - right: L'elemento viene spostato sul lato destro, il contenuto scorre a sinistra
  - none: l'elemento mantiene la sua posizione normale (default)

! *Attenzione: le immagini hanno una dimensione intrinseca e quindi non ci sono problemi ma se si usa questa proprietà con elementi che non hanno una dimensione naturale (es. un paragrafo) bisogna definire anche la proprietà width*

Esempio: immagine senza float

```
<html>
<head>
</head>
<body>
<p><img border="0" src="images/galotti.gif" width="65" height="57">Domenica 27 dicembre
2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione (massimo
25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7...<p>
</body>
</html>
```

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale
laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni. Dai personaggi di Walt
animati che affascinano i ragazzi di tutte le età, ma ancora più affascinante è poter diven
e scoprendo i segreti dell'animazione.

Esempio: immagine con float left e right

```
<html>
<head>
<style type="text/css">
   img.sx { border-width: none; float: left}
   img.dx { border-width: none; float: right}
 </style>
</head>
<body>
<p><img border="0" src="images/galotti.gif" class=sx width="65" height="57">Domenica 27
dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione
(massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7...</p>
<p><img border="0" src="images/galotti.gif" class=dx width="65" height="57">Domenica 27
dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione
(massimo 25 partecipanti) il laborator
</body>
</html>
```

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione (massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni. Dai personaggi di Walt Disney ai Gormiti, dalle Winx a Ben 10, sono tanti i cartoni animati che affascinano i ragazzi di tutte le età, ma ancora più affascinante è poter diventare per un giorno un disegnatore, creando i propri personaggi e scoprendo i segreti dell'animazione. Tutto questo sarà possibile al Museo del Patrimonio industriale con "Making Toons", un laboratorio dedicato ai ragazzi per imparare a fare semplici cartoni animati, approfondendone la scienza e la storia.

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione (massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni. Dai personaggi di Walt Disney ai Gormiti, dalle Winx a Ben 10, sono tanti i cartoni animati che affascinano i ragazzi di tutte le età, ma ancora più affascinante è poter diventare per un giorno un disegnatore, creando i propri personaggi e scoprendo i segreti dell'animazione. Tutto questo sarà possibile al Museo del Patrimonio industriale con "Making Toons", un laboratorio dedicato ai ragazzi per imparare a fare semplici cartoni animati, approfondendone la scienza e la storia.

Esempio con paragrafo floating

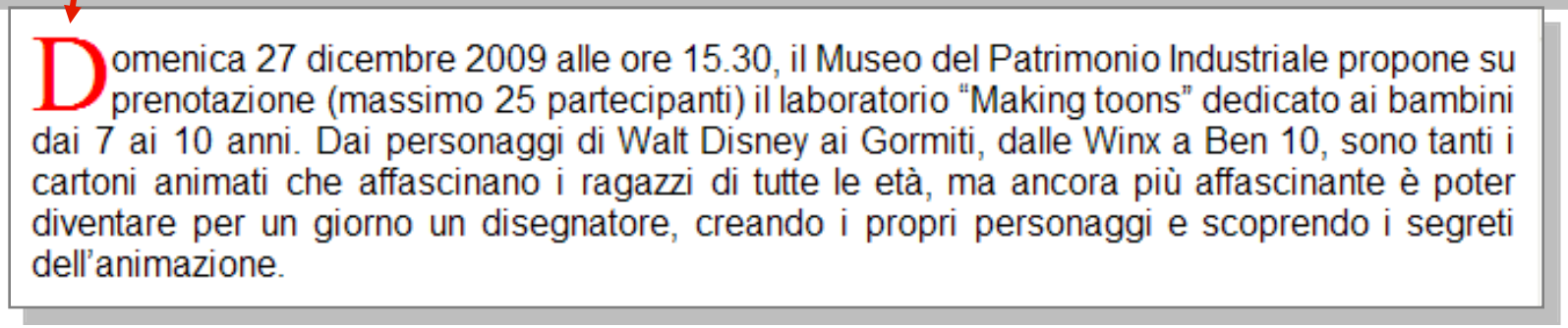
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<style type="text/css">
  img.sx { border-width: none; float: left}
  div.dx {background: #F0F0F0; padding: 4px; float: right; width: 200px}
</style>
</head>
<body>
<p><img border="0" src="images/galotti.gif" class=sx width="65" height="57"><div
class=dx>Come si arriva al museo: prendere l'autobus n. 11 e scendere alla fermata di via
Darwin </div>Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale
propone su prenotazione (massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai
bambini dai 7 ai 10 anni. Dai personaggi di Walt Disney ai Gormiti, dalle Winx a Ben 10,
sono tanti i cartoni animati che affascinano i ragazzi di tutte le età, ma ancora più
affascinante è poter diventare per un giorno un disegnatore, creando i propri personaggi e
scoprendo i segreti dell'animazione.</p>
</html>
```

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione (massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni. Dai personaggi di Walt Disney ai Gormiti, dalle Winx a Ben 10, sono tanti i cartoni animati che affascinano i ragazzi di tutte le età, ma ancora più affascinante è poter diventare per un giorno un disegnatore, creando i propri personaggi e scoprendo i segreti dell'animazione.

Come si arriva al museo: prendere l'autobus n. 11 e scendere alla fermata di via Darwin

Esempio: paragrafo giustificato con capoverso

```
<html>
<head>
<style type="text/css">
  p { text-align: justify; }
  p.primo:first-letter { font-family: Times New Roman; font-size: xx-large; float: left;
                         color: red}
</style>
</head>
<body>
<p class=primo>Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio
Industriale propone su prenotazione (massimo 25 partecipanti) il laboratorio “Making toons”
dedicato ai bambini dai 7 ai 10 anni. Dai personaggi di Walt Disney ai Gormiti, dalle Winx
a Ben 10, sono tanti i cartoni animati che affascinano i ragazzi di tutte le età, ma ancora
più affascinante è poter diventare per un giorno un disegnatore, creando i propri
personaggi e scoprendo i segreti dell’animazione.</p>
<p>Luned&igrave 29 dicembre 2009… </p>
</html>
```



clear

- La proprietà **`clear`** serve a disattivare l'effetto della proprietà float sugli altri elementi che lo seguono, ovvero a impedire che al fianco di un elemento float (dx, sx o entrambi) compaiano altri elementi
  - Il float toglie infatti un elemento dal flusso normale del documento e può quindi capitare che esso venga a trovarsi a fianco di elementi successivi
- Si applica solo agli elementi blocco e non è ereditata
- Valori:
  - none: gli altri elementi possono stare sia a destra che a sinistra dell'elemento
  - left: impedisce il posizionamento a sinistra
  - right: impedisce il posizionamento a destra
  - both: impedisce il posizionamento su entrambi i lati

Esempio senza clear

```
<html>
<head>
<style type="text/css">
  img.sx { border-width: none; float: left }
</style>
</head>
<body>
<p><img border="0" src="images/galotti.gif" class= sx width="65" height="57">Domenica 27
dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione
(massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10
anni.</p>
<p>Dai personaggi di Walt Disney ai Gormiti, dalle Winx a Ben 10, sono tanti i cartoni
animati che affascinano i ragazzi di tutte le età, ma ancora più affascinante è poter
diventare per un giorno un disegnatore, creando i propri personaggi e scoprendo i segreti
dell'animazione. </p>
</html>
```

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimon
laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni.

Dai personaggi di Walt Disney ai Gormiti, dalle Winx a Ben 10, son
più affascinante è poter diventare per un giorno un disegnatore, creando i pro

Esempio con clear: left

```
<html>
<head>
<style type="text/css">
   img.sx { border-width: none; float: left }
   p.cl { clear: left }
</style>
</head>
<body>
<p><img border="0" src="images/galotti.gif" class=sx width="65" height="57">Domenica 27
dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione
(massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10
anni.</p>
<p class=cl>Dai personaggi di Walt Disney ai Gormiti, dalle Winx a Ben 10, sono tanti i
cartoni animati che affascinano i ragazzi di tutte le età, ma ancora più affascinante è
poter diventare per un giorno un disegnatore, creando i propri personaggi e scoprendo i
segreti dell'animazione. <p>
</html>
```

Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimon
laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni.

Dai personaggi di Walt Disney ai Gormiti, dalle Winx a Ben 10, sono tanti i ca
affascinante è poter diventare per un giorno un disegnatore, creando i propri

Posizionamento: position

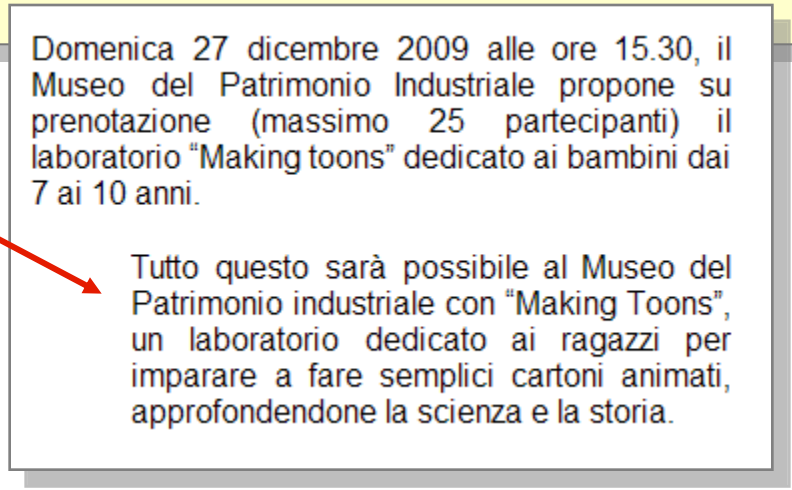
- `position` è la proprietà fondamentale per la gestione della posizione degli elementi, di cui determina la modalità di presentazione sulla pagina
- Non è ereditata e ammette i seguenti valori:
  - static: (default) posizionamento naturale nel flusso
  - absolute: il box dell'elemento viene rimosso dal flusso ed è posizionato rispetto al box contenitore del primo elemento antenato «posizionato», ovvero non static (al limite <html>)
  - relative: l'elemento viene posizionato relativamente al box che l'elemento avrebbe occupato nel normale flusso del documento
  - fixed: posizionamento rispetto al viewport (browser window)

Altre proprietà di posizionamento

- **`left,top,right,bottom`**: coordinate del posizionamento (assoluto o relativo)
- **`visibility`** determina la visibilità e ammette 2 valori:
  - visible: l'elemento è visibile
  - hidden: l'elemento è invisibile ma il suo posto rimane visibile, anche se appare vuoto
- **`z-index`**: CSS gestisce gli elementi come se fossero fogli di carta e questa proprietà permette di stabilire quale sta sopra e quale sta sotto
  - Valori ammessi:
    - auto: lascia al browser la decisione di che ordine attribuire agli elementi
    - valore numerico: più è alto e più l'elemento è in cima al "mucchio di fogli"

Esempio: paragrafo rientrato

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<style type="text/css">
   p { text-align: justify; width: 350 }
   p.rientro { position: relative; left: 50; width: 300 }
</style>
</head>
<body>
<p>Domenica 27 dicembre 2009 alle ore 15.30, il Museo del Patrimonio Industriale propone su prenotazione (massimo 25 partecipanti) il laboratorio "Making toons" dedicato ai bambini dai 7 ai 10 anni. </p>
<p class=rientro>Tutto questo sarà possibile al Museo del Patrimonio industriale con "Making Toons", un laboratorio dedicato ai ragazzi per imparare a fare semplici cartoni animati, approfondendone la scienza e la storia.</p>
</body>
</html>
```



Tabelle

- Alcune proprietà permettono di gestire le tabelle
- **`table-layout`**: non è ereditata e ammette due valori:
  - auto: layout trattato automaticamente dal browser
  - fixed: layout controllato dal CSS
- **`border-collapse`**: definisce il trattamento dei bordi interni e degli spazi fra celle e ammette due valori:
  - collapse: se viene impostato un bordo, le celle della tabella lo condividono
  - separate: se viene impostato un bordo, ogni cella ha il suo, separato dalle altre
- Se si usa separate lo spazio tra le celle e tra i bordi si imposta con la proprietà **`border-spacing`**, che ammette come valore un numero con unità di misura

CSS 3: relazioni con CSS 2.1

- I CSS 2.1 hanno rappresentato un pezzo di storia del Web degli ultimi dieci anni. Hanno aperto la strada ai CSS 3
- Lo sviluppo dei CSS 3 è strettamente correlato con HTML 5
- Come già visto:
  - I CSS 3 estendono le funzionalità CSS 2.1 mantenendone la piena compatibilità
  - A differenza di CSS 2.1, che si presentata come unica grande specifica, CSS 3 è diviso in diversi moduli:
    - CSS level 3 *Media queries* (dal 2012)
    - CSS level 3 *Namespaces* (dal 2011)
    - CSS level 3 *Selectors* (dal 2011)
    - CSS level 3 *Color* (dal 2011)
    - …

CSS 3: principali caratteristiche

- Tra le caratteristiche nuove che contraddistinguono i CSS 3 ricordiamo:
  - Nuovi selettori, pseudo-classi e pseudo-elementi
    - I CSS 3 si avvalgono ora di selettori più potenti che permettono di *selezionare gli elementi in base alla presenza di nuovi token* non contemplati in CSS 2.1 e di *nuovi contesti del DOM*
      - se prima potevamo selezionare un elemento in base alla presenza di un attributo: `p[class]{}`

        ora possiamo selezionarlo anche in base alla presenza di particolari stringhe al suo interno: `p[class^="html"]{}`
      - ancora, prossimo selezionare elementi in base alla loro posizione nel DOM: `p:nth-child(3){}`

        o in base al loro particolare stato nella *user interface*:

        `input:disabled{}`

CSS 3: principali caratteristiche

- Supporto completo a XML
  - I CSS 2.1 non erano in grado di gestire i namespace XML. I CSS 3 ora sono in grado di farlo tramite la regola
    `@namespace element "http://www.sito.it/ns/element/"`
- Gestione completa del layout
  - I CSS 3 consentono ora di avere il pineo controllo di un layout: posizionamento, sfondi multipli, trasparenze, angoli smussati, effetti sul testo, etc.
- CSS per caratteristiche specifiche dei dispositivi
  - I CSS 3 ora consentono di indirizzare gli stili non solo ai dispositivi specifici, ma anche di specificare quali condizioni devono essere soddisfatte per l'applicazione di tali stili tramite le nuove regole
    `@media screen and (max-width: 800px){}`
- Maggiore controllo sulla stampa
  - Con i CSS 3 è ora possibile creare intestazioni, piè di pagina, note in calce e numerazioni automatiche delle pagine per la stampa

HTML 5 e CSS 3: quale conclusione?

- Arrivati a questo punto del programma, ben sappiamo che HTML definisce *struttura/contenuto* di una pagina Web, ovvero la sua parte *fondamentale*!
- Lasciarla in balìa di un *codice che ancora non è recepito come vero standard da parte di tutti i Web browser è comunque un rischio*…
  - utilizzare la specifica HTML 5 su progetti commerciali non è ancora la scelta più giusta, ma nulla vieta di sperimentarlo sui propri siti personali
- Diverso è il discorso sul codice CSS che, per definizione, *definisce la presentazione del contenuto della pagina*!

HTML 5 e CSS 3: quale conclusione?

- Un sito ben strutturato nel suo contenuto, è visibile e comprensibile a prescindere dal CSS impostato o meno: certo non sarà bellissimo… un sito Web senza CSS risulterebbe un po' monotono e noioso, non potremmo riconoscere a colpo d'occhio il sito in cui ci troviamo magari dai colori o dalle immagini di sfondo… *però continueremo a fruire delle informazioni che il sito Web ci offre*!!!
- Questo vuol dire solo una cosa:
  - il CSS è un accessorio
- Sembra brutto a dirlo o anche a pensarlo, soprattutto dal punto di vista di un Web designer o da appassionati di grafica, ma è così! D'altro canto, questa triste verità ci dà l'agio di poter sperimentare i CSS 3 in maniera molto più libera rispetto a HTML5 , perché l'importante è che il contenuto sia fruibile… ☺

HTML 5 e CSS 3: quale conclusione?

- Dunque?! Quale conclusione?!
- La cosa più importante è che impostare lo stile a CSS 3 non vuol dire rinunciare al CSS 2.1
- Per cui la buona pratica sarebbe di creare un «bel sito» con CSS 2.1, compatibile e visibile a tutti i browser attualmente in circolazione, per poi renderlo ancora più bello grazie ai CSS 3, migliorando l'esperienza di navigazione dell'utenza abituata ai browser più moderni (o anche no, utilizzando librerie Javascript)

http://www.csszengarden.com/

- ! *Tanti esempi di CSS sulla medesima struttura HTML*
- Utile per esercitarsi e capire come funziona CSS…

Next: Bibliografia

Riferimenti

- Specifiche ufficiali W3C:

  `http://www.w3.org/TR/CSS2/`
- Ottimo riferimento, con possibilità di fare esercizi:
  `http://www.w3schools.com/css/default.asp`
- Un sito di “CSS estremo” che permette di cambiare radicalmente l’aspetto delle pagine applicando diversi stili:
  `http://www.csszengarden.com/tr/italiano/`