

CSS

PROF. DIOMAIUTA CRESCENZO

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA «*LUIGI VANVITELLI*»

Fogli di stile CSS

- I fogli di stile a cascata** (Cascading Style Sheets = CSS) separano il contenuto dalla presentazione nelle pagine Web
- Il linguaggio HTML serve a definire il contenuto senza tentare di dare indicazioni su come rappresentarlo
- I CSS servono a definire come il contenuto deve essere presentato all'utente
- È prevista la presenza di fogli di stile multipli, che agiscono uno dopo l'altro, in cascata

NOTA! cerca di evitare conflitti

Fogli di stile 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, tablet e smartphone)
- Si può dividere il lavoro fra chi gestisce il contenuto e chi si occupa della parte grafica

Fogli di stile CSS

- 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, tablet, smartphone, ecc.

Compatibilità con i browser

- Il supporto dei vari browser a CSS è complesso e difficile
- Infatti, tutti hanno supportato aspetti diversi ed incompatibili delle caratteristiche di CSS
- <http://www.w3schools.com/css/>
- <http://www.w3schools.com/css3/>

Pagina di esempio

- Creiamo una pagina HTML - denominata swbd.html
- Non ci sono indicazioni su come rappresentare la pagina
- E' stato solo specificato che si tratta di un titolo di primo livello e di un paragrafo
- Il browser userà gli stili standard

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Sistemi Web e Basi di Dati</TITLE>
  </HEAD>
  <BODY>
    <H1>SWBD!</H1>
    <p>Prova con i CSS</p>
  </BODY>
</HTML>
```

Pagina di esempio

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

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

- Collegiamo i due file inserendo il link al CSS nella testata della pagina `swbd.html`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE> Sistemi Web e Basi di Dati </TITLE>
    <LINK rel="stylesheet" href="swbd.css" type="text/css">
  </HEAD>
  <BODY>
    <H1>SWBD!</H1>
    <p>Prova con i CSS</p>
  </BODY>
</HTML>
```

Pagina di esempio - Risultato

- Ecco il risultato nei due casi:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE> Sistemi Web e Basi di Dati </TITLE>
  </HEAD>
  <BODY>
    <H1>SWBD!</H1>
    <p>Prova con i CSS</p>
  </BODY>
</HTML>
```

swbd.html

SWBD !
Prova con i CSS

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE> Sistemi Web e Basi di Dati con CSS</TITLE>
    <LINK rel="stylesheet" href="swbd.css" type="text/css">
  </HEAD>
  <BODY>
    <H1>SWBD!</H1>
    <p>Prova con i CSS</p>
  </BODY>
</HTML>
```

swbd.html

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

swbd.css

SWBD !
Prova con i CSS

Applicazione stili ad una pagina web

- 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> Sistemi Web e Basi di Dati </TITLE>
    <link rel="stylesheet"
          href="swbd.css" type="text/css">
  </HEAD>
  <BODY>
    <H1>SWBD!</H1>
    <p>Prova con i CSS</p>
  </BODY>
</HTML>
```

```
<HTML>
  <HEAD>
    <TITLE> Sistemi Web e Basi di Dati </TITLE>
    <style type="text/css">
      @import url(swbd.css);
    </style>
  </HEAD>
  <BODY>
    <H1>SWBD!</H1>
    <p>Prova con i CSS</p>
  </BODY>
</HTML>
```

Applicazione stili ad una pagina web

- 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> Sistemi Web e Basi di Dati
    </TITLE>
    <style type="text/css"> BODY {
      color: red } H1 { color: blue
    }
    </style>
  </HEAD>
  <BODY>
    <H1>SWBD!</H1>
    <p>Prova con i CSS</p>
  </BODY>
</HTML>
```

```
<HTML>
  <HEAD>
    <TITLE> Sistemi Web e Basi di Dati </TITLE>
  </HEAD>
  <BODY style="color: red">
    <H1 style="color: blue"> SWBD!</H1>
    <p>Prova con i CSS</p>
  </BODY>
</HTML>
```

Quale scegliere?

- È 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 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 retrocompatibilità con i vecchi browser

CSS – regole e 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

- 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 { ... }`
- **Identifieri:** si applicano agli elementi che presentano l'attributo `id="nome_id"`
`#nome_id { ... }`

Selettori

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

tipo_elemento.nome_classe { ... }

tipo_elemento#nome_id { ... }

- Negli esempi della pagina precedente la mancanza di un tipo_elemento prima di . o di # sottintendeva la presenza del selettore universale *

.nome_classe → * . nome_classe

#nome_id → * . #nome_id

Selettori

- **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

- **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 elementi 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

- **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
- **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

- **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 due metodi differenti:
 - In forma esadecimale `#RRGGBB`
 - Tramite la funzione `rgb(rosso,verde,blu)`

Ereditarietà

- È un meccanismo simile per certi aspetti all'ereditarietà nella programmazione 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)

Ereditarietà

- L'elemento `<p>Prova con i CSS</p>` non ridefinisce il colore del testo e quindi eredita da `<body>`: viene mostrato in rosso
- L'elemento `<H1>SWBD!</H1>` ridefinisce lo stile e quindi appare in blu

swbd.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Sistemi Web e Basi di Dati</TITLE>
    <LINK rel="stylesheet" href="swbd.css"
          type="text/css">
  </HEAD>
  <BODY>
    <H1>SWBD!</H1>
    <p>Prova con i CSS</p>
  </BODY>
</HTML>
```

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

swbd.css

SWBD!
Prova con i CSS

Conflitti di stile

- Quando si applicano 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
- **errore o gestione disordinata dei CSS**

Cascade – Risoluzione di conflitti

- 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}
```

Cascade – Risoluzione di 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 uguale specificità e origine vince quella fornita per ultima.

Le dichiarazioni esterne hanno la precedenza rispetto a qualsiasi dichiarazione interna

Clausola !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> }

Background

- 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
- 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

Font-family

- La proprietà per 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

- 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**, poi 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**

Font-size

- La proprietà **font-size** permette di definire le dimensioni del testo
- 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
 - con l'unità **ex**: proporzione rispetto all'altezza del corpo della x minuscola del font corrente
 - in **percentuale**

Font-size

- Qual è il miglior modo di definire il corpo di un carattere?
- In teoria sullo schermo sarebbe bene usare i pixel, ma alcuni browser danno problemi
- W3C consiglia 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 e font-style

- La proprietà **font-weight** definisce il peso del carattere (es: grassetto)

- La proprietà **font-style** permette di definire varianti del testo rispetto al normale
 - **normal**: valore di default (tondo)
 - **italic**: testo in corsivo
 - **oblique**: testo obliquo, simile a italic

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**

Font – proprietà sintetica

- 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**

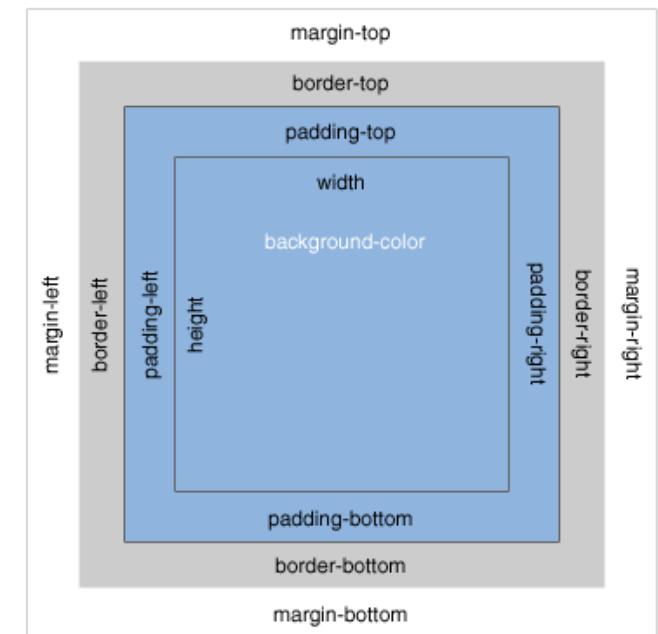
- Es: **p {font: italic normal bold 10px Arial,Serif;}**
- I font di sistema permettono di adattare le pagine all'aspetto del sistema operativo, sono 6 valori:
 - **caption**: font usato per bottoni e combo-box
 - **icon**: font usato per il testo delle icone
 - **menu**: carattere usato nei menu delle 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

Allineamento e decorazione del testo

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

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**



Box model

- **Area del contenuto**: testo, immagine, ecc. di cui è possibile definire altezza e larghezza
- **Padding**: spazio vuoto tra il contenuto e il bordo dell'elemento
- **Border**: di cui possiamo definire colore, stile e spessore
- **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** 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

- **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
- **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)

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, padding-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

```
<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>
```

Liste

- CSS definisce alcune proprietà che agiscono sulle liste puntate e numerate , più precisamente sugli elementi delle liste
- 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

- **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

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)

Float

- Con **float** è possibile estrarre un elemento dal normale flusso del documento e spostarlo su uno dei lati del suo elemento contenitore
- La proprietà non è ereditata
- In HTML questa possibilità era riservata 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)

→ for!

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

Posizionamento

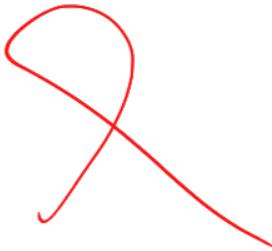
- **position** è la proprietà fondamentale per la gestione della posizione degli elementi, di cui determina la modalità di presentazione sulla pagina
- Non è ereditata e ammette i seguenti valori:
 - **static**: (default) posizionamento naturale nel flusso
 - **absolute**: il box dell'elemento viene rimosso dal flusso ed è posizionato rispetto al box contenitore del primo elemento antenato «posizionato»
 - **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)

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

Potrà non primo piano mentre
grado con gli index.

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 Media Queries

- Tra le **caratteristiche introdotte da CSS3 ci sono le media queries.**
- Si possono usare
 - All'interno dell'attributo media del tag link, per importare un certo foglio di stile solo se la query è soddisfatta, oppure
 - Direttamente nei fogli di stile racchiudendo insiemi di regole all'interno di parentesi graffe, precedute dalla at-rule @media
- Una media query è composta da un **media type** (screen, print, ecc.) messa in **AND** con una serie di media features, il cui supporto può variare nei browser.
 - orientation: [portrait | landscape]
 - width: <misura>, min-width: <misura>, max-width: <misura>
- E' possibile mettere in OR più media queries separandole con una virgola, ad esempio

```
@media screen and (min-width: 300px),
screen and (orientation: landscape) {...}
```

Responsive design

- Per i browser “normali”
 - Design liquido, con ampiezze percentuali (possibilmente anche per le immagini). Un design liquido rende il layout robusto anche per i dispositivi che non supportano ancora le media queries.
- Per i tablet:

Ad esempio: @media only screen and (min-width: 768px) and (max-width: 959px)

- Eliminare padding e spaziature “creative”, diminuire leggermente la dimensione del carattere, ecc.
- Per i cellulari:

Ad esempio: @media only screen and (max-width: 767px)

- Design “fixed”, ad esempio di 320 pixel, oppure inserire una min-width sugli elementi principali per evitare che si riducano troppo.
- Eliminare le colonne linearizzandone il contenuto, nascondere gli elementi secondari (parte di header e footer, ecc.), mostrare menu più compatti.

Esempi

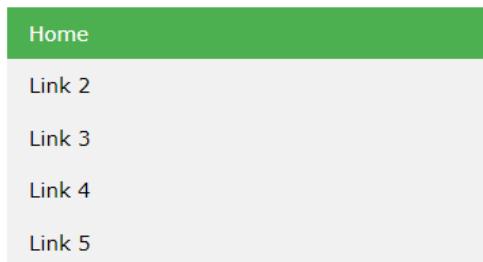
https://www.w3schools.com/w3css/w3css_navigation.asp

W3.CSS Navigation Bars

◀ Previous

Next ▶

Vertical:



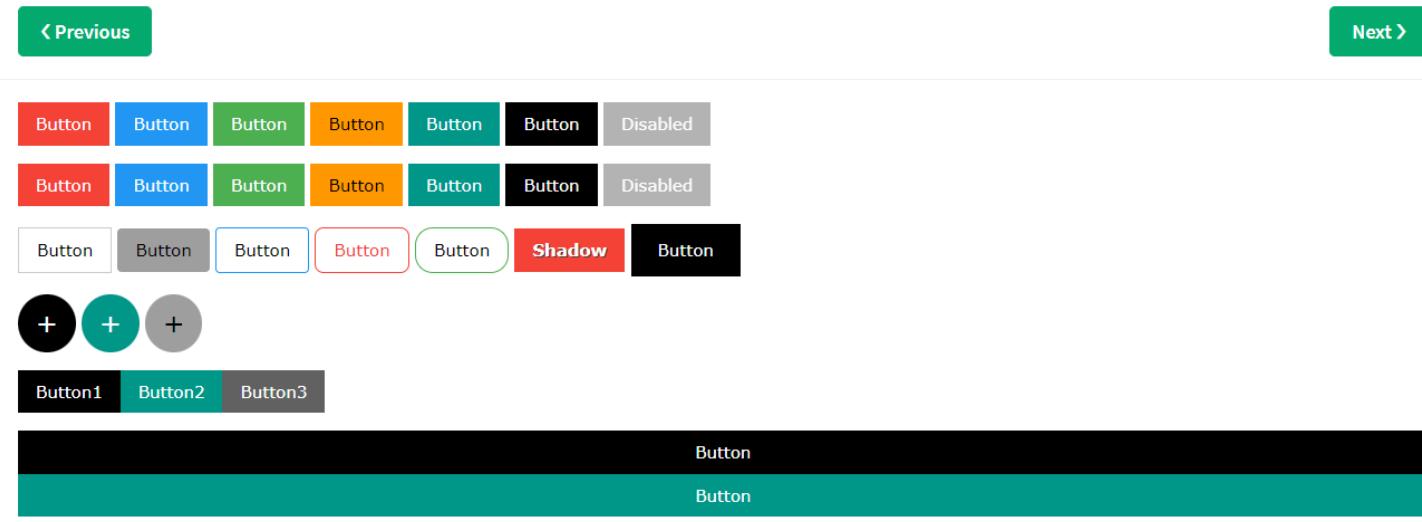
Horizontal:



Esempi

https://www.w3schools.com/w3css/w3css_buttons.asp

W3.CSS Buttons



Esempi

https://www.w3schools.com/w3css/w3css_cards.asp

W3.CSS Cards

◀ Previous

Next ▶



Header

Some text.. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Footer

Esempi

This is Sofia

Sofia on Fire

Making the Web!

Making the Web!

Making the Web!

Making the Web!

Esempi

https://www.w3schools.com/w3css/w3css_notes.asp

W3.CSS Notes

◀ Previous

Next ▶

London

London is the most populous city in the United Kingdom, with a metropolitan area of over 9 million inhabitants.

London

London is the most populous city in the United Kingdom, with a metropolitan area of over 9 million inhabitants.

London

London is the most populous city in the United Kingdom, with a metropolitan area of over 9 million inhabitants.

Esempi

https://www.w3schools.com/w3css/w3css_lists.asp

W3.CSS Lists

◀ Previous

Next ▶

	Mike Web Designer	X
	Jill Support	X
	Jane Accountant	X

Esempi

https://www.w3schools.com/w3css/w3css_input.asp

Input Form

Name

Email

Subject

Milk
 Sugar
 Lemon (Disabled)

Male
 Female
 Don't know (Disabled)

Send ➤

Riferimenti

- Specifiche ufficiali W3C:

<http://www.w3.org/TR/CSS/>

- Ottimo riferimento, con possibilità di fare esercizi:

<http://www.w3schools.com/css/default.asp>

<https://www.w3schools.com/w3css/>