# CSS

## Introduzione, sintassi e concetti di base

# Premessa

- Tra tutte le tecnologie web è forse quella più odiata ...
- Rimane un linguaggio FONDAMENTALE: tutti i siti contengono del CSS.
- Non solo! Anche molte app!
- È necessario imparare le basi del linguaggio e comprendere il suo funzionamento.
- Successivamente, vi mostreremo anche strumenti che vi faciliteranno nello sviluppo del CSS.

# L'origine

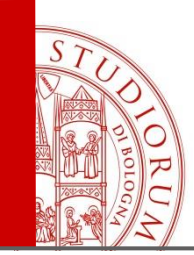
- Il CSS (Cascading Style Sheets), proposto da Bert Bos e Håkon Lie, risponde all'esigenza di una tecnologia per la resa grafica degli ipertesti.
- Hanno lo scopo fondamentale di **separare contenuto** e **presentazione** nelle pagine Web.
  - **HTML** serve per definire il **contenuto** senza fornire indicazioni su come presentarlo.
  - **CSS** serve per definire **come** il contenuto deve essere presentato
- Prevista ed incoraggiata la presenza di fogli di stile **multipli**, che **agiscono uno dopo l'altro**, in **cascata**, per indicare come un documento HTML deve essere visualizzato.

# Vantaggi

- Lo stesso contenuto può essere presentato in modi diversi:
  - http://www.csszengarden.com/219/
  - http://www.csszengarden.com/220/
- Lo stesso contenuto può essere presentato *correttamente* su dispositivi diversi (pc, tablet e smartphone) o su media diversi (video o carta)
- Si può dividere il lavoro fra chi gestisce il contenuto e chi la presentazione
- Si riduce il tempo di scaricamento delle pagine
- Non c'è la necessità di usare estensioni come Flash

# Versioni

- CSS level 1 (W3C Rec. 1996, revisione 2008): *linguaggio di formattazione visiva* per specificare caratteristiche tipografiche e di presentazione per gli elementi di un documento HTML.
- CSS level 2 (W3C Rec. 1998) e CSS level 2.1 (W3C Rec. 2011, revisione 2014): introduce il supporto per media multipli e un layout più sofisticato
- CSS level 3 (W3C Working Draft), alcune sezioni sono già recommendation (Colors, Selectors, Fonts), altre sono ancora in alto mare…
- CSS level 4 in discussione, prevede una profonda ri-modularizzazione del linguaggio



# Supporto da parte dei browser

- Il supporto dei vari browser a CSS è complesso e difficile: tutti i browser hanno supportato e supportano in modo diverso i CSS
  - Nessun browser ha mai supportato completamente Level 1, anche se già i primi browser che supportavano CSS avevano meccanismi per il posizionamento assoluto degli oggetti nella pagina Web (che fa parte di Level 2)
  - Ancora oggi nessun browser supporta completamente Level 2
  - Differenze sostanziali nel supporto alle regole di Level 3

# Verifica Supporto da parte dei browser

- È possibile controllare le funzionalità supportate dai diversi browser nelle seguenti pagine
  - https://caniuse.com
  - https://www.w3schools.com/css/
  - https://www.w3schools.com/css/css3_borders.asp

# Usare CSS con HTML

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

MAI USARLO 1. Posizionato presso il tag di riferimento (foglio di stile ***inline***, attraverso l'attributo **`style`**)

MAI USARLO 2. Posizionato nel tag **`<style>`** (foglio di stile ***interno***, nell'**`header`** del documento)

MAI USARLO 3. Importato dal tag **`<style>`** (foglio di stile ***esterno importato***, nell'**`header`** del documento)

 4. Indicato dal tag **`<link>`** (foglio di stile ***esterno***, nell'**`header`** del documento)

# 1 Foglio di stile *inline*

1 - Posizionato nel tag di riferimento (foglio di stile ***inline***)

```
<html>
  <head>
      <title>Monsters and Co.</title>
  </head>
  <body>
    <header style="color:blue;">
      <h1>Monsters and Co.</h1>
     </header>
    <section>
      <p>Monsters and Co. (Monsters, Inc.) &egrave; un film
   d'animazione della Pixar, del 2001 diretto da Pete Docter,
   Lee Unkrich e David Silverman.</p>
      </section>
  </body>
</html>
```

# Foglio di stile *interno*

2 - Posizionato nel tag **<style>**

```
<html>
  <head>
  <title>Monsters and Co.</title>
   <style type="text/css">
      header { color: blue; }
    </style>
  </head>
  <body>
    <header>
       <h1>Monsters and Co.</h1>
     </header>
    <section>
       <p>Monsters and Co. (Monsters, Inc.) &egrave; un film
   d'animazione della Pixar, del 2001 diretto da Pete Docter, Lee
   Unkrich e David Silverman.</p>
       </section>
  </body>
</html>
```

# Foglio di stile *esterno importato*

3 - Importato dal tag **<style>**

```
<html>
  <head>
    <title>Monsters and Co.</title>
    <style type="text/css">
      @import url(style.css);
    </style>
  </head>
  <body>
     <header>
        <h1>Monsters and Co.</h1>
     </header>
    <section>
        <p>Monsters and Co. (Monsters, Inc.) &egrave; un film
 d'animazione della Pixar, del 2001 diretto da Pete Docter, Lee
 Unkrich e David Silverman.</p>
        </section>
  </body>
</html>
```

style.css

```
header
{color:blue;}

…
```

# Foglio di stile *esterno* *[CONSIGLIATO]*

4 - Indicato dal tag **<link>**

```
<html>
  <head>
    <title>Monsters and Co.</title>
    <link type ="text/css"
      rel  ="stylesheet" href="style.css"/>
  </head>
  <body>
    <header>
      <h1>Monsters and Co.</h1>
    </header>
    <section>
      <p>Monsters and Co. (Monsters, Inc.) &egrave; un
  film d'animazione della Pixar, del 2001 diretto da Pete
  Docter, Lee Unkrich e David Silverman.</p>
    </section>
  </body>
</html>
```

style.css

```
header
{color:blue;}

…
```

# Sintassi

- Una regola CSS ha la seguente forma

  **`Selettore { Proprietà: Valore;}`**

- Un **selettore** consente di specificare un elemento o un insieme di elementi dell'albero HTML al fine di associarvi delle caratteristiche.

  Es: **`header`**, **`section`**, **`footer`**

- Una **proprietà** è una caratteristica di stile assegnabile ad un elemento.

  Es: **`background-color`**, **`width`**, **`height`**

- I **valori** dipendono ovviamente dalla proprietà.

  Es: *red*, *60%*, *100px*

# I selettori

## Tipologie di selettori

- **Selettore universale** (*): fa match con qualsiasi elemento
- **Selettore di tipo** (**E**): fa match con gli elementi E (seleziona i tipi di elementi HTML)

```
body{ font-family: Arial; font-size: 12 pt; }
header { font-size: 18 pt; }
section { font-size: 10 pt; }
```

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

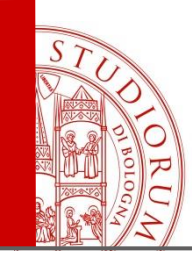
Fratelli

```
section p { font-size: 10 pt; }
```
corrisponde a tutti i <p> all'interno di <section>
Seleziona ogni elemento F che sia figlio, nipote, pronipote, … di un E

```
p>strong { color: red; }
```
Seleziona ogni elemento F che sia figlio immediato di un E con > non selezioni i nipoti oltre il primo livello

```
header+h1 { font-size: 11 pt; }
```
Seleziona l'elemento F solo se segue immediatamente un elemento E con lo stesso genitore.

```
img ~ p { margin-top: 0.5em; }
```
Seleziona tutti gli elementi F che seguono E (non necessariamente immediatamente), purché condividano lo stesso genitore.



- id è pensato per identificare in modo univoco un elemento (id deve essere unico all'interno di un documento)
- class serve per raggruppare elementi "simili" o appartenenti a uno stesso ruolo semantico/presentazionale (class può essere assegnato a più elementi contemporaneamente)

2

# I selettori

- E[foo]: Seleziona tutti gli elementi E che possiedono l'attributo foo (indipendentemente dal valore).
- E[foo="bar"]: Seleziona gli elementi E con foo esattamente uguale a "bar".
- E[foo~="bar"]: Seleziona gli elementi E il cui attributo foo contiene la parola bar, all'interno di una lista separata da spazi.
- E[foo^="bar"]: Seleziona gli elementi E in cui l'attributo foo inizia con la stringa "bar".

- **Selettori di attributi** (**E[foo], E[foo="bar"], E[foo~="bar"], E[foo^="bar"]**): Fanno match con gli elementi E che possiedono l'attributo specificato o che ha un valore particolare.

```
a[name] { color: red; }
```

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

- .bar seleziona qualsiasi elemento che abbia la classe bar.
- E.bar è un selettore composto che colpisce solo gli elementi di tipo E con classe bar.
- #bar seleziona l'elemento (unico nella pagina) che ha id="bar".
- E#bar seleziona solo se il tag è di tipo E e ha quell'ID.

elemento h1 con classe spiegazione

```
h1.spiegazione { font-size: 24 px; }
```

ogni elemento con classe spiegazione

```
.spiegazione { font-size: 12 px; }
```

ogni elemento p con id note1

```
p#note1 { font-size: 9 px; }
```

ogni elemento con id note5

```
#note5 { color: red; }
```

# Nota sui selettori di attributi

Funzionano anche nel caso in cui l'attributo dichiarato non sia valido (codice HTML non valido rispetto alla grammatica dichiarata):

```
<!DOCTYPE html>
<html lang="it">
<head>
    <style>
        div:first-of-type { margin-bottom: 5px;}
        div { padding: 10px; display: inline-block; }
        div[id] { background-color: lime; }
        div[attributo-inesistente] { background-color: yellow;}
    </style>
</head>
<body>
   <div id="div1"><p>Div con attributo valido.</p></div>
   <div attributo-inesistente><p>Questo con attributo non valido.</p>
   </div>
</body>
</html>
```

# 3 I selettori

- **Selettori di pseudo-classi** (**E:link, E:visited, E:active, E:hover, E:focus, E:enabled, E:checked, E:lang(c)**):
  - **link, visited**: vero se l'elemento E è un link non ancora visitato o un link già visitato.
  - **hover, active, focus**: vero se sull'elemento E passa sopra il mouse, il mouse è premuto o il controllo è selezionato per accettare input.
  - **enabled, checked**: vero se elemento E è abilitato o «checked».
  - **lang(c)**: vero se l'elemento ha selezionata la lingua c.
- **,: raggruppamento di selettori** (selettori diversi possono usare lo stesso blocco se separati da virgola)

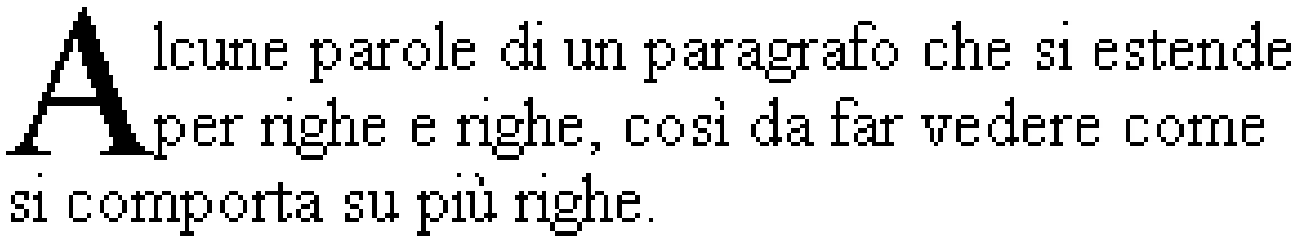
# I selettori

- **Selettori di pseudo-classi strutturali** (**E:first-child, E:nth-child(n), E:nth-last-child(n), E:first-of-type, E:nth-of-type(n), E:only-of-type, E:empty**):
  - **first-child**: elemento E che è il primo figlio di suo padre.
  - **nth-child(n)**: elemento E che è l'n-esimo figlio di suo padre.
  - **nth-last-child(n)**: elemento E che è l'n-esimo figlio di suo padre a partire dall'ultimo.
  - **first-of-type**: elemento E che è il primo figlio di suo padre di quel tipo.
  - **nth-of-type(n)**: elemento E che è l'n-esimo figlio di suo padre di quel tipo.
  - **only-of-type**: elemento E che è l'unico figlio di suo padre di quel tipo. (se non é l'unico, non si applica lo stile)
  - **empty**: elemento E che è vuoto.

# I selettori

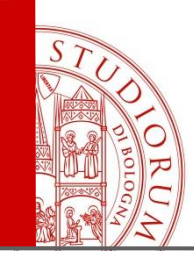
- **Selettori di pseudo-elementi** (**E:first-line  E:first-letter  E:before  E:after**): Vengono attivati in corrispondenza di certe parti degli elementi E.
  - **before, after**: vero prima e dopo il contenuto dell'elemento E.
  - **first-line**: vero per la prima riga dell'elemento E.
  - **first-letter**: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

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



# Valori

- Grandezze: numeri seguiti da unità di misura
- Numeri interi e reali (il punto è il separatore dei decimali)
- Unità di misura:
  - Relative:
    - ***em***: relativa alla dimensione del font in uso (es: se il font ha corpo 12pt, 1em varrà 12pt, 2em varranno 24pt, …)
    - ***px***: relativi al dispositivo di output e alle impostazione dell'utente
  - Assolute:
    - ***in***: pollici (1in = 2.54cm)
    - ***cm***: centimetri
    - ***mm***: millimetri
    - ***pt***: punti tipografici (1/72 di pollice)
    - ***pc***: pica (12pt)



# Valori (2)

- Percentuali: percentuale del valore che assume la proprietà stessa nell'elemento padre
- URL assoluti o relativi **`url(path)`**
- Stringhe
- Colori: possono essere specificati in diversi modi come esadecimale (**`#RRGGBB`**) o con una keyword (**`black`**, **`silver`**, **`white`**, **`red`**, …)

# Conflitti di stile

- Nell'applicare il CSS possono nascere dei conflitti, ovvero ad uno stesso elemento sono applicate delle regole i cui valori sono in conflitto.
- Esempio:

```
div#provaID{ background-color: red;}
div.provaClasse{ background-color: blue;}
div{background-color: green; }

<div id='provaID'
class='provaClasse'></div>
```

- Di che colore sarà lo sfondo del `<div>`?

# La cascata – Ordinamento regole

- Le dichiarazioni vengono ordinate in base ai seguenti fattori (ordinati dal più «fino» al meno importante):

1 – Media

Solo le dichiarazioni appartenenti al media corrente vengono considerate; quelle in @media non corrispondenti al dispositivo attivo vengono ignorate.

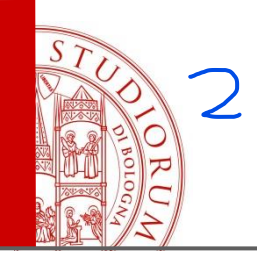
2 – Importanza di una dichiarazione

3 – Origine della dichiarazione

4 – Specificità del selettore

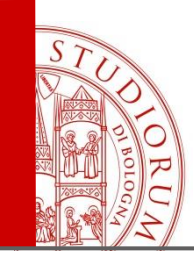
5 – Ordine delle dichiarazioni

Infine, se tutti i fattori precedenti sono in parità (stessa origine, stessa importanza, stessa specificità), vince l'ultima dichiarazione nel foglio (o – in caso di più fogli – nell'ordine di caricamento degli stili).



# Importanza della dichiarazione

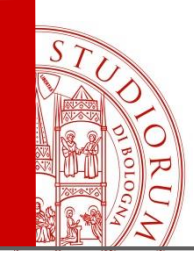
- È possibile aggiungere ad una dichiarazione la keyword **`!important`**.
- Una regola contenente questa keyword avrà precedenza sulle altre, indipendentemente da origine, specificità e ordine delle dichiarazioni.
- Questa può essere utilizzata dagli utenti per imporre alcune regole per loro importanti (molto utile in caso di utente con necessità specifiche: disabilità visive, daltonismo, ecc).
- Esempio: `p { font-size: 18pt !important }`



3

# Origine della dichiarazione

- Un foglio di stile può avere 3 origini differenti, qui riportate in ordine decrescente di importanza:
  - ***Author:*** l'autore delle pagine fornisce i fogli di stile del documento specifico
  - ***User***: l'utente può fornire un ulteriore foglio di stile per indicare regole di proprio piacimento. Tipicamente è una funzione del browser
  - ***User Agent***: il browser definisce (esplicitamente o implicitamente, codificandole nel software) le regole di default per gli elementi dei documenti



4

# Specificità del selettore

- La specificità di un selettore è data da una quadrupla **xywz** dove:
  - **x:** 1 se la dichiarazione è nell'attributo style, 0 altrimenti.
  - **y:** numero di ***id*** specificati nel selettore. (gli id si specificano con #)
  - **w:** numero di classi, attributi e pseudo-classi specificati nel selettore. (quindi .bar , [] , : )
  - **z:** numero di elementi e di pseudo-elementi specificati nel selettore.
- A parità di Media, Importanza e Origine, avrà precedenza la regola con specificità più alta.

# Specificità del selettore - Esempi

- `li` ? 0001 (GIUSTO)
- `nav ul li:first-line` ? 0013 (non é 13 perché ci sono 3 elementi ed un pseudo-elemento (non é una pseudoclasse))
- `nav.menu ul.sec li` ? 0023 (GIUSTO)
- `nav ul li a[href='/home']` ? 0014 (GIUSTO)
- `nav#menu ul.sec li#st a` ? 0214 (GIUSTO)
- `style="li a"` ? 1002 (GIUSTO)

# Specificità del selettore - Esempi

- `li` /* x=0 y=0 w=0 z=1 => 1 */
- `nav ul li:first-line` /* x=0 y=0 w=0 z=4 => 4 */
- `nav.menu ul.sec li` /* x=0 y=0 w=2 z=3 => 23 */
- `nav ul li a[href='/home']` /* x=0 y=0 w=1 z=4 => 14 */
- `nav#menu ul.sec li#st a` /* x=0 y=2 w=1 z=4 => 214 */
- `style="li a"` /* x=1 y=0 w=0 z=2 => 1002 */

# Prima domanda


**DOMANDA 1:**

Qual è la specificità del seguente selettore

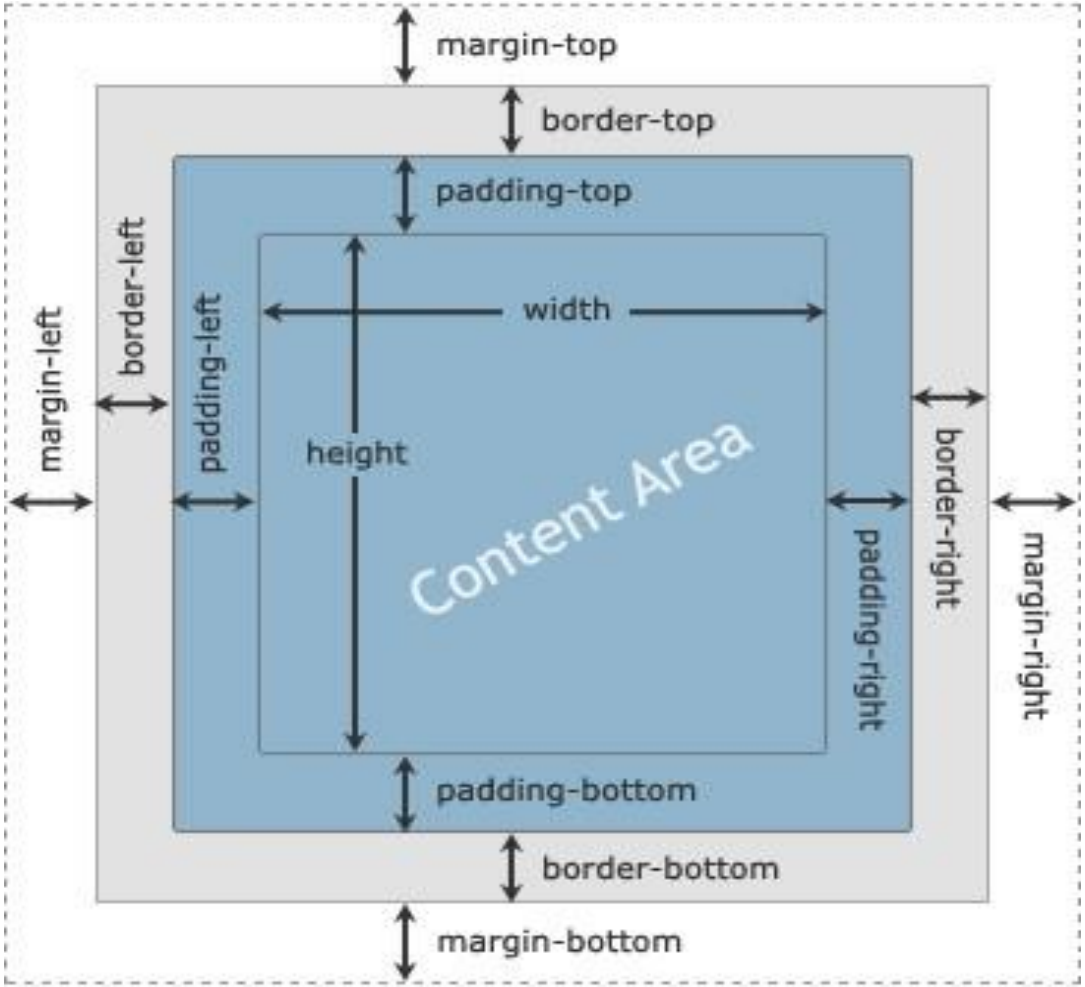
```
aside#left p.first a img[src='logo.png']
```

- ❑ 1114
- ☒ 124 [1 id, (1 attributo e 1 classe) = 2, 4 elementi]
- ❑ 34
- ❑ 214

# Box Model

- Ogni elemento è definito da una scatola (box) all'interno della quale si trova il contenuto.
- La visualizzazione di un documento con CSS avviene identificando lo spazio di visualizzazione di ciascun box presente nella pagina.

# Box Model




https://bit.ly/2ZCj5uX

# Contenuto

- È possibile definire le dimensioni del contenuto con le proprietà **`width`** e **`height`.**
- È possibile definire:
  - una dimensione minima con le proprietà **`min-width`** e **`min-height`**
  - una dimensione massima con le proprietà **`max-width`** e **`max-height`**

- **NB:** Solitamente si specifica SOLO la larghezza e NON la l'altezza. In questo modo l'altezza di un elemento viene determinata dal suo contenuto.

# Dimensioni del contenuto

- Cosa succede nel caso in cui vengano specificate larghezza e altezza di un elemento MA il suo contenuto richiede più spazio?
- È possibile gestire questa situazione con la proprietà **`overflow`** che può avere i seguenti valori:
  - **`visible`**: il contenuto eccedente viene mostrato
  - **`hidden`**: il contenuto eccedente viene nascosto
  - **`scroll`**: vengono mostrare le barre di scorrimento per visualizzare il contenuto eccedente
  - **`auto`**: il contenuto eccedente viene mostrato in base alle impostazioni del browser

Il contenuto eccedente viene ritagliato, e le barre di scorrimento compaiono solo se realmente necessarie. Se tutto il contenuto rientra, il comportamento è equivalente a visible (tranne che crea comunque un nuovo contesto di formattazione).

# Margin

- Permette di impostare lo spazio tra un elemento e gli altri elementi della pagina.
- Quattro proprietà singole: **`margin-top`**, **`margin-right`**, **`margin-bottom`**, e **`margin-left`**
- Possibili valori :
  - Valore numerico con unità di misura.
  - Valore in percentuale.
- È possibile utilizzare la proprietà abbreviata **`margin`**:

```
p{margin: 5px 7px 8px 10px} /*top right bottom left*/
p{margin: 5px 7px 6px } /*top right-left bottom*/
p{margin: 5px 10%} /*top/bottom right-left*/
p{margin: 5px } /*all*/
```

# Margin

- Nel caso in cui due elementi siano allineati **orizzontalmente**, la distanza tra i due è data dalla somma dei due margini (il margine destro del primo elemento e il margine sinistro del secondo).

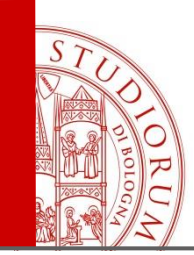

- Nel caso in cui due elementi siano allineati **verticalmente**, si ha il cosiddetto **margin collapsing**: la distanza tra i due è data dal valore massimo fra il margine inferiore del primo elemento e quello superiore del secondo.
- **NB**: stesso comportamento che ritroviamo anche in Word.

# Padding

- Permette di impostare lo spazio fra il contenuto e il bordo. Al contrario dei margini, il **`padding`** ha lo stesso colore di sfondo dell'elemento.
- Quattro proprietà singole: **`padding-top`**, **`padding-right`**, **`padding-bottom`**, e **`padding-left`.**
- Possibili valori:
  - Valore numerico con unità di misura
  - Valore in percentuale
- Come per margin, anche per padding esiste la proprietà abbreviata **`padding`**.

# Border

- Permette di impostare lo spessore, lo stile e il colore di ognuno dei quattro bordi.
- Esistono tre proprietà singole per ognuno dei quattro bordi (dodici in totale): **`border-`*`position`*`-width`**, **`border-`*`position`*`-style`** e **`border-`*`position`*`-color`** (dove *position* può essere **`top`**, **`right`**, **`bottom`**, **`left`**)
- Esistono 3 tipi di proprietà sintetiche:
  - **`border-top`**, **`border-right`**, **`border-bottom`**, **`border-left`**
  - **`border-width`**, **`border-style`**, **`border-color`**
  - **`border`**.



# Border - Valori

- Spessore:
  - Valore numerico con unità di misura
  - Keyword (**`thin`**, **`medium`**, **`thick`**)
- Stile:
  - **`none`** o **`hidden`**: nessun bordo
  - **`solid`**: intero
  - **`dotted`**: a puntini
  - **`dashed`**: a trattini
  - **`double`**: doppio
  - **`groove`**, **`ridge`**, **`inset`**, **`outset`**: effetti tridimensionali
- Colore

# Dimensioni del Box

- La larghezza complessiva dei box è data dalla seguente formula (che considera il box-model):

```
margin-left + border-left-width +
padding-left + width + padding-right +
border-right-width + margin-right
```

- Se **`width`** non è impostata, viene determinata in automatico dal browser.
- Per l'altezza complessiva dei box vale un discorso analogo MA bisogna tenere in considerazione il **margin collpasing**.

# Seconda domanda


**DOMANDA 2:**

Quale di queste forme abbreviate non è equivalente alle altre:

- ❑ `margin: 20px 10px 20px 10px;`
- ☒ `margin: 20px 10px 10px;` (perché ha bottom diverso)
- ❑ `margin: 20px 10px;`
- ❑ `margin: 20px 10px 20px;`

# Posizionamento

- La disposizione degli elementi all'interno della pagina è una delle questione più complesse e che provocano più frustrazione negli sviluppatori!

# Posizionamento

- In base a cosa vengono disposti gli elementi?
- Quanti comportamenti diversi possiamo osservare?
- Identifichiamo 2 comportamenti diversi.

# Posizionamento – Comportamento 1

- Relativo agli elementi **`<h1>`**, **`<h2>`**, **`<p>`**, **`<div>`**.
- Larghezza:
  - Se non specificata occupano il 100% di quella del padre.
  - È possibile specificare un valore con la proprietà **`width`**.
- Altezza:
  - L'altezza dipende dal contenuto dell'elemento.
  - È possibile specificare un valore con la proprietà **`height`**.
- A prescindere dalla larghezza, gli elementi sono disposti verticalmente, formando una nuova riga.
- Questi elementi sono chiamati **elementi di blocco**.

# Posizionamento – Comportamento 2

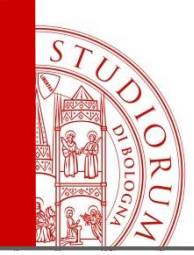
- Relativo agli elementi ***a***, ***strong***, ***em***, ***span***.
- Larghezza:
  - La larghezza dipende dal contenuto dell'elemento.
  - Non è possibile specificare un valore con la proprietà ***width***.
- Altezza:
  - L'altezza dipende dal contenuto dell'elemento.
  - Non è possibile specificare un valore con la proprietà ***height***.
  - È possibile specificare l'altezza della linea con la proprietà ***line-height***.
- Gli elementi adiacenti sono disposti orizzontalmente.
- Questi elementi sono chiamati **elementi di linea**.

# Posizionamento

- Problema molto dibattuto e con diverse soluzioni «truccologiche»: ***perfect centering*** …
  (si risolve in modo molto semplice con i **`flexbox`**)

# Display

- La proprietà **`display`** determina il tipo di elemento (e il relativo comportamento). Oltre a **`inline`** e **`block`**, questa proprietà può assumere i seguenti valori:
  - **`none`**: l'elemento non viene visualizzato.
  - **`inline-block`***:* l'elemento può assumere dimensioni esplicite (come gli elementi blocco), ma si disporrà orizzontalmente (come gli elementi inline) e non verticalmente. Con inline-block l'elemento genera un block box (in cui width, height, margin, padding e border vengono rispettati come in un blocco), ma si dispone orizzontalmente con il flusso del testo, esattamente come un elemento inline, senza forzare un'interruzione di riga
  - **`list-item`**: per fare in modo che un elemento si comporti come un **`<li>`**.
  - **`grid`***:* trasforma un elemento in un grid container.
  - **`flex`***:* trasforma un elemento in un flex container.



# Display (2)

- Esistono anche valori per trasformare elementi in parti di una tabella: **`table`**, **`inline-table`**, **`table-cell`**, **`table-row`**, **`table-row-group`**, **`table-column`**, **`table-column-group`**, **`table-header-group`**, **`table-footer-group`**, **`table-caption`**.
- Altri valori raramente usati: **`contents`**, **`run-in`**, **`marker`**, **`compact`**.

# Layout multi colonna liquido - Display

- Un layout liquido è un layout in cui la grandezza della pagina dipende dalla finestra del browser, adattandosi a tutte le risoluzioni.
- Può essere realizzato usando la proprietà **`display`** rendendo i contenitori delle 3 colonne di tipo **`inline-block`** e definendo la larghezza delle colonne in percentuale.
- Esempio: si vuole realizzare un layout a 3 colonne dove:
  - La colonna a sinistra contiene il menù e deve occupare il 15% della pagina.
  - La colonna a destra è una semplice sidebar e deve occupare il 20% della pagina.
  - La colonna centrale contiene un articolo e deve occupare il 65% della pagina.

# Layout con Display - Riassunto

- Le colonne devono essere elementi ibridi **`inline-block`**.
- Gli elementi di linea ~~sono solitamente allineati in basso~~ , di default, si allineano in basso. Se le colonne sono di altezze diverse (molto probabile) è necessario specificare un allineamento a partire dall'alto usando la proprietà **`vertical-align`** con il valore **`top`**.
- Le tre colonne devono occupare in totale al massimo 100% tra **`width`**, **`margin`** e **`padding`**, altrimenti l'ultima andrà a capo. Con box-sizing: border-box, il padding e il border sono già inclusi nella % di width. Se superi il 100%, la colonna in eccesso va a capo automaticamente (o andate a capo)
- NON devono esserci spazi nel codice HTML tra una sezione e l'altra. Altrimenti l'ultima colonna andrà a capo.
- Siccome i bordi non possono essere specificati in percentuale (e non avrebbe neanche senso farlo), è necessario usare la proprietà **`box-sizing`** con valore **`border-box`** per fare in modo che la grandezza del bordo (E DEL PADDING!) sia inclusa nella larghezza.

# Float

- Abbiamo visto che gli elementi di blocco vengono disposti verticalmente, uno sotto l'altro.
- Float consente di estrarre un elemento dal normale flusso del documento e lo sposta su un lato, a destra o a sinistra (rispetto al suo contenitore).
- Gli elementi appartenenti al normale flusso del documento circonderanno gli elementi «floating».

# Layout multi colonna liquido - Float

- Un altro modo per realizzare un layout multi colonna liquido consiste nell'utilizzare la proprietà **`float`** e definendo la larghezza delle colonne in percentuale.
- Stesso esempio di prima: si vuole realizzare un layout a 3 colonne dove:
  - La colonna a sinistra contiene il menù e deve occupare il 15% della pagina.
  - La colonna a destra è una semplice sidebar e deve occupare il 20% della pagina.
  - La colonna centrale contiene un articolo e deve occupare il 65% della pagina.

# Layout con Float - Riassunto

- Le colonne laterali devono essere float, quella centrale no.
- La colonna centrale deve avere dei margini laterali almeno delle dimensioni delle colonne laterali (maggiore se si vogliono distanziare le colonne).
- Le tre colonne devono occupare in totale al massimo 100%, altrimenti ci saranno delle sovrapposizioni.
- Abbiamo visto come gli **<h2>** presenti in **<nav>** e **<aside>** NON sono soggetti al **margin collapse**, in quanto con **float** sono fuori dal normale flusso della pagina, al contrario di article.
- Vale lo stesso discorso di prima per i bordi anche se l'effetto non è che l'ultimo box va a capo ma che c'è una sovrapposizione.

La proprietà clear serve a “interrompere” il flusso dei float precedenti, costringendo l’elemento a comparire sotto (e non più accanto) ai box flottanti.

# Clear

Quando un elemento porta float: left; o float: right;, viene estratto dal normale flusso del documento e “incollato” al lato sinistro o destro del suo contenitore. Tutti gli elementi successivi nel markup, se non hanno clear, scorrono e si dispongono intorno ad esso

- Abbiamo detto che **`float`** consente di estrarre un elemento dal normale flusso del documento e lo sposta su un lato, a destra o a sinistra. Può quindi capitare che questo venga a trovarsi a fianco di elementi successivi.
- La proprietà **`clear`** serve a disattivare l’effetto della proprietà **`float`** sugli elementi che lo seguono, ovvero a impedire che al fianco di un elemento floating compaiano altri elementi.
- Valori:
  - **`none`**: float consentito su entrambi i lati.
  - **`left`**: impedisce il posizionamento a sinistra. — l’elemento viene spostato sotto tutti i float impostati a sinistra che lo precedono.
  - **`right`**: impedisce il posizionamento a destra. — idem, ma per i float a destra.
  - **`both`**: impedisce il posizionamento su entrambi i lati. — l’elemento viene costretto sotto tutti i float, sia di sinistra sia di destra

# Position

- Un altro modo per gestire la disposizione degli elementi nella pagina è la proprietà **`position`** che consente di specificare il posizionamento dell'elemento rispetto al flusso del documento.
- Possibili valori (spesso usati):
  - **`static`**: valore di default, l'elemento è disposto secondo il normale flusso del documento.
  - **`fixed`**: usando questo valore il box dell'elemento viene sottratto al normale flusso del documento. Il box non scorre con il resto del documento, ma rimane fisso. (anche se scrollo la pagina, l'elemento rimane sempre nello stesso punto della pagina, cioé segue lo scroll)

# Position

- Possibili valori (**DA EVITARE**):
  - **`relative`**:
    - L'elemento NON viene rimosso dal flusso del documento.
    - A partire dalla posizione che avrebbe occupato, è possibile specificare lo spostamento con le proprietà **`top`**, **`right`**, **`bottom`** e **`left`** (accettano anche valori negativi).
  - **`absolute`**:
    - L'elemento viene rimosso dal flusso del documento.
    - Il posizionamento avviene rispetto al primo elemento antenato che ha un posizionamento diverso da **`static`** (se non esiste viene usata la radice **`<html>`**).
    - Il posizionamento è specificato sempre attraverso le proprietà **`top`**, **`right`**, **`bottom`** e **`left`**.

# Position

- In caso di elementi sovrapposti, è possibile gestire quale elemento deve essere visualizzato «sopra» con la proprietà **`z-index`**. Verrà visualizzato l'elemento con **`z-index`** maggiore (che si andrà a sovrapporre agli elementi con **`z-index`** inferiore).

- **NB**: **`z-index`** funziona SOLO con elementi che non abbiano **`static`** come posizione.

In CSS l'ereditarietà è il meccanismo per cui, quando un elemento non ha un valore esplicito per una certa proprietà, prende il valore calcolato che ha sul suo elemento padre

# Ereditarietà

Ogni elemento, per poter essere visualizzato, parte da uno stile di base definito dal browser (user-agent stylesheet).

- Per poter essere visualizzato, ogni elemento DEVE avere uno stile «di base». Un elemento privo di stile non può essere rappresentato.
- Lo stile può essere applicato:
  - Direttamente: con l'attributo **`style`** o con regole. (foglio esterno)
  - Indirettamente: l'elemento eredità lo stile dal padre.
- Non tutte le proprietà sono soggette ad ereditarietà, ad esempio:
  - **`display`**: questa dipende intrinsecamente dall'elemento stesso.
  - **`background`**: è sempre trasparente.
  - Proprietà relative al box model…
  - …
- È possibile forzare l'ereditarietà usando come valore **`inherit`**.

Se una proprietà non ha un valore diretto, se ne eredita il calcolato dal genitore.

```
<style>
  /* Il body ha colore blu */
  body { color: blue; }
  /* Nessuna regola su <p>, perciò eredita il color dal body */
  p { font-size: 1rem; }
</style>
<body>
  <p>Testo in blu (perché ereditato)</p>
</body>
```

# Conclusioni

- In questa lezione abbiamo introdotto il CSS, contestualizzandolo e illustrando i concetti alla base del suo funzionamento: sintassi, il concetto di cascata, il box model e il meccanismo di ereditarietà.
- Poi abbiamo esaminato solo come disporre gli elementi all'interno della pagina.
- Nella prossima lezione vi illustreremo i diversi stili associabili agli elementi HTML (sfondo, colori, testo, immagini, …) e infine vi parleremo di come creare layout responsive.

# Riferimenti

- Standard completi:
  - CSS1, https://www.w3.org/TR/CSS1/
  - CSS2, http://www.w3.org/TR/CSS2
  - CSS3, https://www.w3.org/TR/2001/WD-css3-roadmap-20010523/

# Approfondimenti

- «CSS3 Guida completa per lo sviluppatore», Peter Gasston, 2011. Disponibile in biblioteca
- Esercizi sui selettori https://flukeout.github.io/

# Domande?