

Tecnologie Web e Laboratorio: CSS



Fabio Buttussi

Dipartimento di Matematica, Informatica e Fisica
Università degli Studi di Udine

Materiale nelle slide

► Fonti:

- libro di testo (HTML&CSS. Design and build websites. Jon Duckett, Wiley, 2011)
- materiale del dott. Stefano Burigat
- materiale su CSS3 fornito dalla dott.ssa Emanuela Pitassi
- W3C Schools
(<https://www.w3schools.com/css>)
- esempi dal materiale di supporto del libro
(<http://www.htmlandcssbook.com/code-samples/>)
- esercizi ad hoc per il corso (su Teams e Materiale Didattico)



Struttura

- ▶ Introduzione
- ▶ Selettori
- ▶ Proprietà per lo stile
- ▶ Proprietà per l'impaginazione
- ▶ Layout
- ▶ Responsive design

Introduzione

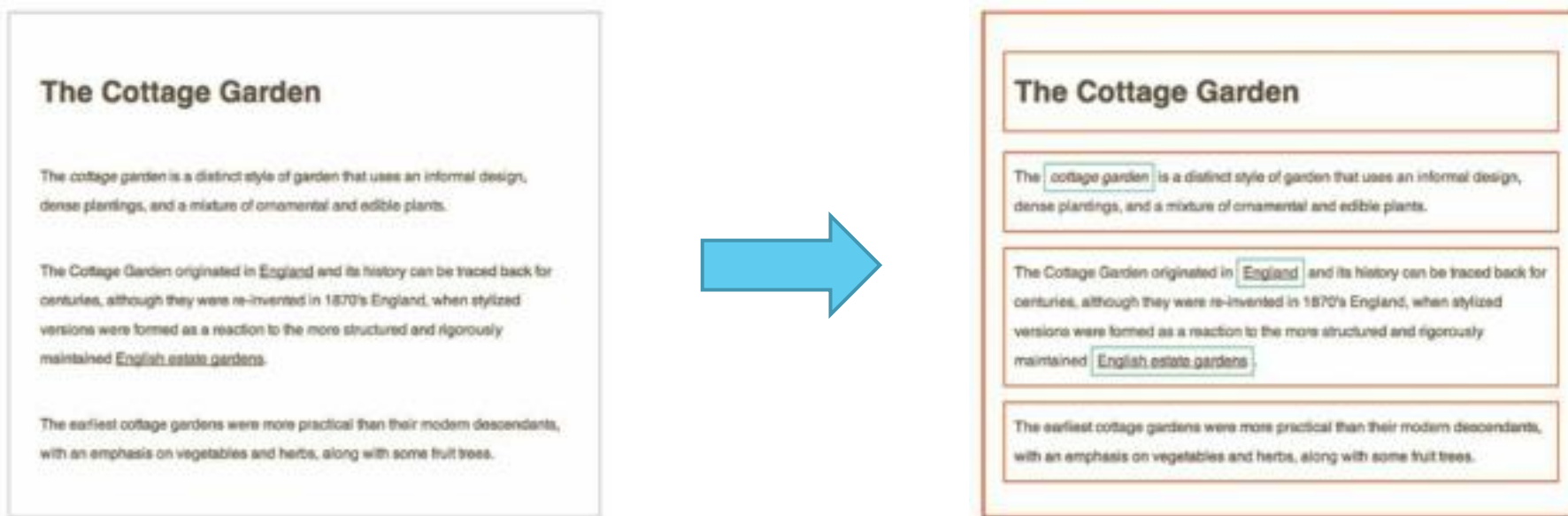
- ▶ XML / HTML -> organizzano i contenuti in modo strutturato
- ▶ CSS -> **specifica come i contenuti devono apparire**
- ▶ CSS = Cascading Style Sheets = Fogli di stile a cascata
- ▶ È quindi necessario:
 - ▶ **Identificare** i contenuti per cui specificare l'aspetto
 - ▶ **Specificare le proprietà** che si vogliono applicare ai contenuti identificati

Capire come funziona

Per capire CSS, immaginiamo dei **box** invisibili attorno ad ogni elemento HTML.

CSS permette di scrivere **regole** per controllare l'aspetto dei box e del loro contenuto.

Nell'esempio, gli elementi block e inline sono rispettivamente bordati in rosso e verde.



Regole (1)

- ▶ Selettore -> permette di identificare gli elementi
- ▶ Dichiarazione -> permette di specificare le proprietà



Regole (2)

- ▶ Possiamo applicare la stessa dichiarazione a più elementi separandoli con la ,
- ▶ Ogni dichiarazione contiene coppie proprietà: valore;
- ▶ Possiamo applicare più proprietà agli stessi elementi separandole con ;

```
h1, h2, h3 {  
    font-family: Arial;  
    color: yellow;  
}
```



PROPERTY VALUE

Cosa possiamo personalizzare

- ▶ Stile:
 - ▶ Font,
 - ▶ Allineamento,
 - ▶ Colore e sfondo,...
- ▶ Impaginazione:
 - ▶ Altezza e larghezza,
 - ▶ Posizione,
 - ▶ Margini,...

CSS3

- ▶ CSS3 è l'ultimo standard ed è retro-compatibile con le versioni precedenti
- ▶ [In questa tabella di Jens Meiert](#) sono illustrati i cambiamenti in termini di proprietà introdotte dai CSS 3 (125 nuove proprietà, per un totale di 245)
- ▶ Per essere aggiornati in tempo reale sull'avanzamento dei lavori del CSSWG (*Cascading Stylesheet Workin Group*) [si può fare riferimento alla relativa pagina](#) che elenca tutte le specifiche e le bozze proposte del gruppo di lavoro
- ▶ Per sapere cosa è supportato dai vari browser (e versioni): <https://caniuse.com/>






Vendor prefix

- ▶ I prefissi dei browser sono un modo per i produttori di browser di aggiungere nuove funzionalità durante i periodi di test
- ▶ I prefissi vengono utilizzati per aggiungere nuove funzionalità che potrebbero non far parte delle specifiche CSS finali
- ▶ Il prefisso per Safari e Chrome è **-webkit**.
 - ▶ Ad es, per specificare il raggio del bordo in vecchie versioni di Chrome e Safari, si utilizzava la seguente sintassi:

`-webkit-border-radius: 24px;`

Vendor prefix

- ▶ I principali prefissi sono i seguenti

Browser	Vendor Prefix
 Firefox	-moz-
 Safari	-webkit-
 Chrome	-webkit-
 Opera	-o-
 Internet Explorer	-ms

<style> o <link>?

- ▶ Utilizzando il tag <style> o la proprietà style di un elemento è possibile includere le regole CSS direttamente nell'HTML
- ▶ Utilizzando il tag <link> è possibile specificare il collegamento ad un foglio di stile CSS esterno
- ▶ La seconda opzione permette di riusare il CSS in più pagine, fare manutenzione più facilmente, usare più file CSS per proprietà diverse,...

```
<html>
  <head>
    <title>Using Internal CSS</title>
    <style type="text/css">
      body {
        font-family: arial;
        background-color: rgb(185,179,175);}
      h1 {
        color: rgb(255,255,255);}
    </style>
  </head>
```

```
<html>
  <head>
    <title>Using External CSS</title>
    <link href="css/styles.css" type="text/css"
      rel="stylesheet" />
  </head>
```

Esempi

- ▶ Analizziamo i file:
 - ▶ [Esempi CSS\Introduzione\thinking-inside-the-box-borderless.html](#)
 - ▶ [Esempi CSS\Introduzione\thinking-inside-the-box.html](#)
 - ▶ [Esempi CSS\Introduzione\example.html](#)
 - ▶ [Esempi CSS\Introduzione\example.css](#)

Selettori (1)

- ▶ Ripasso:
 - ▶ I selettori permettono di identificare gli elementi
 - ▶ È possibile usare più selettori separati da , nella stessa regola
- ▶ Vari tipi di selettori permettono di identificare gli elementi in modi diversi
- ▶ Attenzione: i selettori sono case-sensitive
- ▶ Per gli esempi useremo:
 - ▶ [Esempi CSS\Selettori\css-selectors.html](#)
 - ▶ [Esempi CSS\Selettori\css\selectors.css](#) (le parti tra /* */ sono commenti)

Selettori (2)

Selettore	Significato	Esempio
UNIVERSAL SELECTOR	Applies to all elements in the document	<code>* {}</code> Targets all elements on the page
TYPE SELECTOR	Matches element names	<code>h1, h2, h3 {}</code> Targets the <h1>, <h2> and <h3> elements
CLASS SELECTOR	Matches an element whose class attribute has a value that matches the one specified after the period (or full stop) symbol	<code>.note {}</code> Targets any element whose class attribute has a value of note <code>p.note {}</code> Targets only <p> elements whose class attribute has a value of note

Selettori (3)

Selettore

ID SELECTOR

Significato

Matches an element whose id attribute has a value that matches the one specified after the pound or hash symbol

Esempio

```
#introduction {}
```

Targets the element whose id attribute has a value of introduction

CHILD SELECTOR

Matches an element that is a direct child of another

```
li>a {}
```

Targets any <a> elements that are children of an element (but not other <a> elements in the page)

DESCENDANT SELECTOR

Matches an element that is a descendent of another specified element (not just a direct child of that element)

```
p a {}
```

Targets any <a> elements that sit inside a <p> element, even if there are other elements nested between them

Selettori (4)

Selettore	Significato	Esempio
ADJACENT SIBLING SELECTOR	Matches an element that is the next sibling of another	<code>h1+p {}</code> Targets the first <p> element after any <h1> element (but not other <p> elements)
GENERAL SIBLING SELECTOR	Matches an element that is a sibling of another, although it does not have to be the directly preceding element	<code>h1~p {}</code> If you had two <p> elements that are siblings of an <h1> element, this rule would apply to both

Alt 126

Selettori di attributi (1)

Selettore	Significato	Esempio
EXISTENCE	<code>[]</code> Matches a specific attribute (whatever its value)	<code>p[class]</code> Targets any <code><p></code> element with an attribute called <code>class</code>
EQUALITY	<code>[-]</code> Matches a specific attribute with a specific value	<code>p[class="dog"]</code> Targets any <code><p></code> element with an attribute called <code>class</code> whose value is <code>dog</code>
SPACE	<code>[~=]</code> Matches a specific attribute whose value appears in a space- separated list of words	<code>p[class~="dog"]</code> Targets any <code><p></code> element with an attribute called <code>class</code> whose value is a list of space-separated words, one of which is <code>dog</code>

Selettori di attributi (2)

| = per parola intera

Selettore

PREFIX

Significato

[^=]

Matches a specific attribute whose value begins with a specific string

Esempio

p[attr^="d"]

Targets any <p> element with an attribute whose value begins with the letter "d"

SUBSTRING

[*=]

Matches a specific attribute whose value contains a specific substring

p[attr*"do"]

Targets any <p> element with an attribute whose value contains the letters "do"

SUFFIX

[\$=]

Matches a specific attribute whose value ends with a specific string

p[attr\$"g"]

Targets any <p> element with an attribute whose value ends with the letter "g"

Pseudo-classi relative a stati

- ▶ I selettori possono essere combinati con le **pseudo-classi** che aggiungono attributo fittizio a cui applicare specifiche proprietà:
 - ▶ **:link** imposta lo stile dei link non visitati
 - ▶ **:visited** imposta lo stile dei link visitati
 - ▶ **:hover** imposta lo stile per gli elementi quando l'utente ci è sopra
 - ▶ **:focus** imposta lo stile degli elementi quando hanno il focus
 - ▶ **:active** imposta lo stile degli elementi quando sono attivati dall'utente
- ▶ Questo qui sopra è l'ordine consigliato
- ▶ Esempi:
 - ▶ [Esempi CSS\Selettori\Link Visited.html](#)
 - ▶ [Esempi CSS\Selettori\Hover Active Focus.html](#)

Pseudo-classi strutturali

- ▶ Le pseudo-classi strutturali consentono di selezionare elementi difficili da raggiungere con selettori semplici
 - ▶ **E:root**
 - ▶ **E:first-child**
 - ▶ **E:nth-child()**
 - ▶ **E:nth-last-child()**
 - ▶ **E:last-child**
 - ▶ **E:only-child**
 - ▶ **E:nth-of-type()**
 - ▶ **E:nth-last-of-type()**
 - ▶ **E:first-of-type**
 - ▶ **E:last-of-type**
 - ▶ **E:only-of-type**
 - ▶ **E:empty**
- ▶ NB: Dato un qualsiasi oggetto all'interno del DOM, se l'oggetto contiene degli elementi figli, l'indice dei figli inizia da 1 e non da 0.
- ▶ Ad esempio:

```
<ul>  
  <li>List Item 1</li>    (Posizione 1)  
  <li>List Item 2</li>    (Posizione 2)  
  <li>List Item 3</li>    (Posizione 3)  
  <li>List Item 4</li>    (Posizione 4)  
  <li>List Item 5</li>    (Posizione 5)  
</ul>
```

E:nth-child() e formule

- ▶ E:nth-child() può anche essere utilizzato con delle formule
- ▶ Es., per colorare i testi delle righe di una tabella in maniera alternata:
 - ▶ `tr td {color: black}`
 - ▶ `tr:nth-child(odd) td {color: red}`
- ▶ E:nth-child(an+b) permette di selezionare un elemento che nell'albero del documento è il figlio (an+b)-esimo del proprio padre (a e b possono assumere valori interi positivi, negativi e 0).
- ▶ Es., per assegnare un colore diverso ogni tre righe della tabella, iniziando a contare dalla seconda riga:
 - ▶ `tr:nth-child(3n+2) td {color: red}`

Ulteriori pseudo-classi

- ▶ **E:target** consente di selezionare un elemento E della pagina che corrisponde ad un indirizzo di riferimento.
- ▶ Es., data una pagina con delle ancore al proprio interno, la pseudo-classe E:target permette di assegnare uno stile all'elemento di destinazione dell'ancora nel momento in cui è selezionato
 - ▶ [Esempi CSS\Selettori\Target.html](#)
- ▶ **E:not** identifica gli elementi di tipo E che non coincidono con il selettore contenuto nel :not
- ▶ E:enabled, E:disabled, E:checked ed altre ancora su https://www.w3schools.com/css/css_pseudo_classes.asp

Pseudo-elementi

- ▶ Anche gli pseudo-elementi si combinano ai selettori con :, ma a partire da CSS3 si preferisce usare ::, per distinguerli dalle pseudo-classi
- ▶ Si chiamano così perché creano degli elementi fittizi a cui applicare proprietà personalizzate
- ▶ Alcuni esempi utili:
 - ▶ **::first-letter** identifica la prima lettera degli elementi selezionati
 - ▶ **::first-line** identifica la prima linea degli elementi selezionati
 - ▶ **::before** e **::after** generano ed inseriscono contenuto prima o dopo l'elemento

Cascading

- ▶ Cosa succede se due regole selezionano lo stesso elemento?
 - ▶ Se il selettore è lo stesso, vince l'**ultima regola**
 - ▶ Se un selettore è più specifico dell'altro, vince il **selettore più specifico**
 - ▶ Se una proprietà ha **!important**, vince su quelle senza
- ▶ Possiamo creare prima regole più generiche per più elementi e poi regole specializzate per specifici elementi
- ▶ Possiamo anche usare più file CSS: ad esempio, uno per le regole generali di tutto il sito più uno per pagina con regole specifiche della pagina

Ereditarietà

- ▶ Alcune proprietà (come font-family) sono **automaticamente ereditate** dagli elementi figli
- ▶ Altre proprietà (come background-color) **non** vengono automaticamente ereditate
- ▶ Si può forzare l'ereditarietà dall'elemento contenitore dichiarando la proprietà come **inherit**

```
body {  
  font-family: Arial, Verdana, sans-serif;  
  color: #665544;  
  padding: 10px;}  
.page {  
  border: 1px solid #665544;  
  background-color: #efefef;  
  padding: inherit;}
```

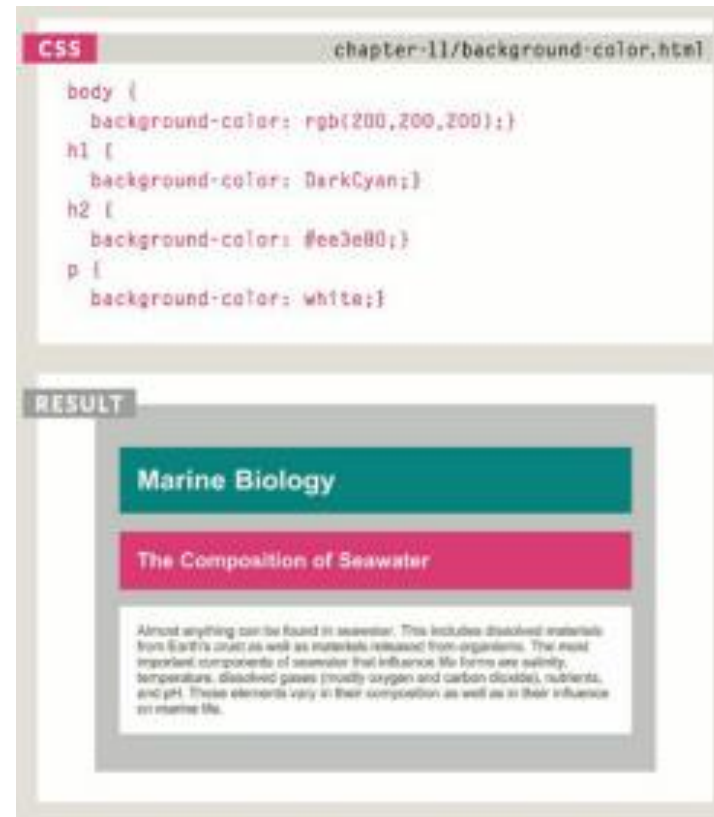
Colore in primo piano

- ▶ Il colore in primo piano (**foreground**) si specifica con la proprietà **color** in diversi modi:
 - ▶ Nome del colore, usando uno dei 140 nomi predefiniti (https://www.w3schools.com/cssref/css_colors.asp)
 - ▶ Valore RGB, indicando quanto rosso, verde e blu da 0 a 255 in `rgb(r,g,b)`
 - ▶ Codice esadecimale, indicando quanto rosso, verde e blu in esadecimale preceduti da `#`

```
/* color name */
h1 {
  color: DarkCyan;}
/* hex code */
h2 {
  color: #ee3e80;}
/* rgb value */
p {
  color: rgb(100,100,90);}
```

Colore di sfondo

- ▶ Il colore di sfondo (background) si specifica con la proprietà **background-color** con le stesse modalità viste per il colore in primo piano
- ▶ Il colore di sfondo riempie il box dell'elemento
- ▶ Nessun colore specificato = trasparente, ma può variare con browser e impostazioni utente



Colore HSL

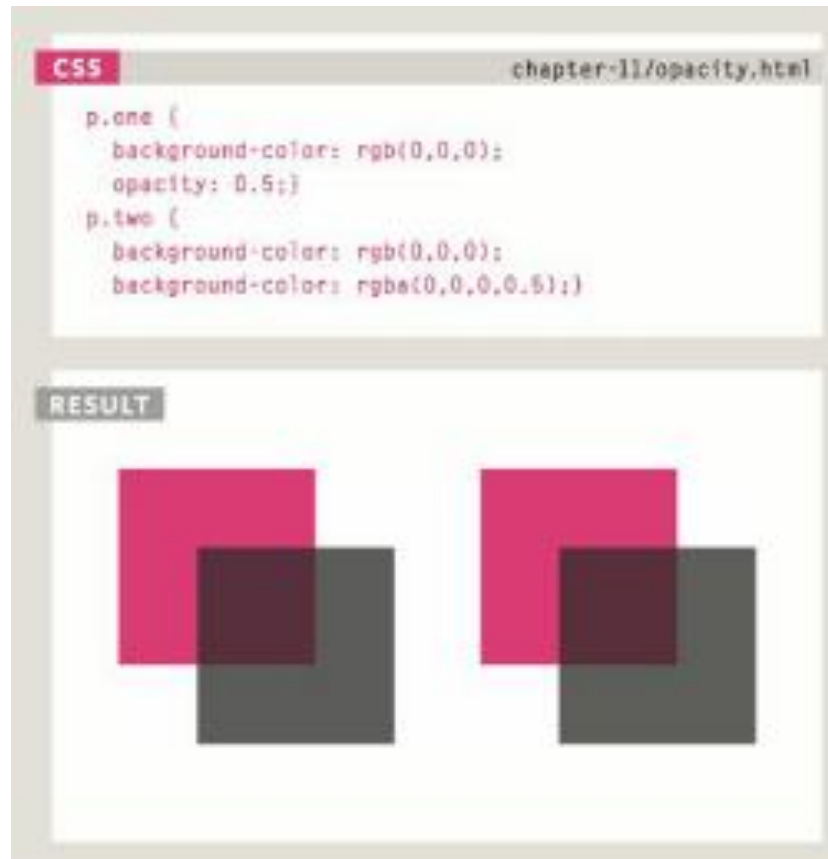
- ▶ CSS3 permette di specificare i colori anche indicando
 - ▶ Hue, ovvero la tonalità di colore specificata come angolo di un cerchio di colori
 - ▶ Saturation, ovvero la saturazione, inversa alla percentuale di grigio (100% = colore pieno e 0% = grigio)
 - ▶ Lightness, ovvero la luminosità come percentuale di bianco (0% = nero, 100% = bianco)
- ▶ Attenzione: lightness non è brightness come in HSB/HSV (https://en.wikipedia.org/wiki/HSL_and_HSV)



```
body {  
  background-color: #C8C8C8;  
  background-color: hsl(0,0%,78%);}
```

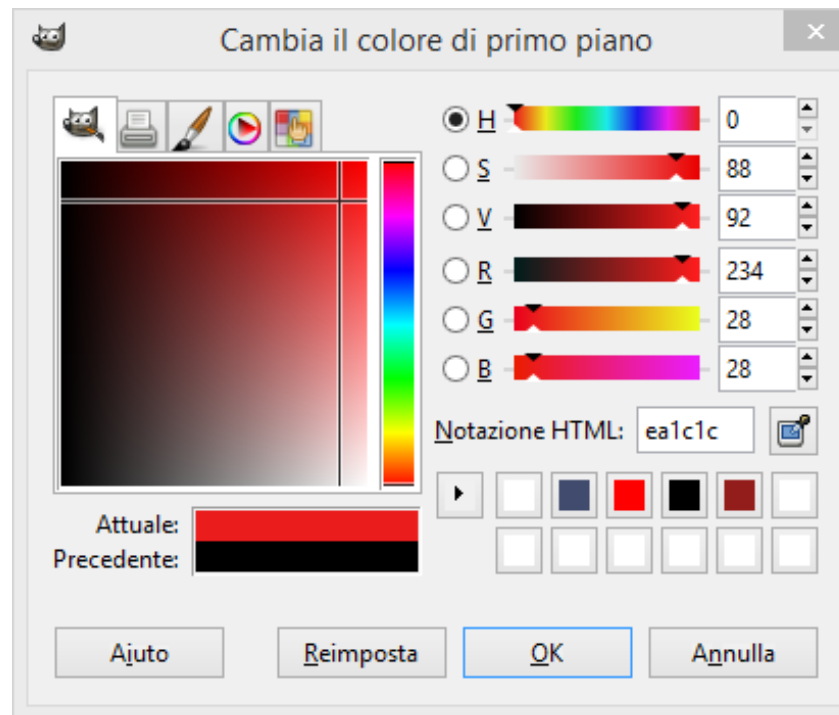
Trasparenza / opacità

- ▶ CSS3 ha introdotto la trasparenza di un elemento (e dei suoi figli) con la proprietà **opacity** tra 0 e 1
- ▶ Si può anche specificare solo l'opacità di un colore (non dei figli) specificando il valore alpha con **rgba** o **hsla**
- ▶ Esempio modificato: [Esempi CSS\Colori\opacity_mod.html](#)



Strumenti utili per gestire i colori

- ▶ Gimp o altri software di grafica
- ▶ ColorZilla
(<https://www.colorzilla.com/>) o altre estensioni per prelevare il colore da pagine Web
- ▶ Convertitori di colori
(https://www.w3schools.com/colors/colors_converter.asp)
- ▶ Calcolatrice, es. $cd = 12 * 16 + 13 = 205$; $171 \% 16 = 10 = a$, $171 \text{ Mod } 16 = 11 = b$, $171 = ab$
- ▶ Per HSL:
https://www.w3schools.com/colors/colors_hsl.asp



Considerazioni di usabilità sui colori



- ▶ Attenzione alla scelta dei colori!
- ▶ Contrasto basso -> difficoltà di lettura in generale, critico in caso di disturbi della vista, critico su monitor di bassa qualità o controluce (es. su mobile)
- ▶ Alto contrasto -> facilità di lettura, non ottimale su testi lunghi
- ▶ Contrasto medio -> ottimale su testi lunghi
- ▶ Se si usa testo chiaro su sfondo scuro, aumentare spazio tra le righe

Immagini in background (1)

- ▶ A differenza del tag HTML `` per inserire immagini come contenuto, CSS offre proprietà per includere immagini come sfondi, separatori, etc.
- ▶ **background-image** permette di aggiungere un'immagine di sfondo ad un qualsiasi elemento (il valore è il path espresso con `url()`)
- ▶ È sempre consigliabile definire un colore di background di tonalità simile a quella dell'immagine, che verrà applicato sotto l'immagine e quando c'è un problema nel caricarla
- ▶ **background-repeat** determina come l'immagine debba essere replicata:
 - ▶ **repeat** (il default)
 - ▶ **repeat-x** (replica orizzontalmente su una sola riga)
 - ▶ **repeat-y** (replica verticalmente su una sola colonna)
 - ▶ **no-repeat** (nessuna replica)
- ▶ Esempio: [Esempi CSS\Colori\immagini_sfondo.html](#)

Immagini in background (2)

- ▶ **background-position** permette di specificare la posizione dell'immagine all'interno dell'elemento, specificando un valore per la posizione orizzontale e uno per quella verticale
- ▶ I valori possono essere definiti in diversi modi:
 - ▶ parole chiave **left**, **right**, **top**, **bottom**, e **center** (se un solo valore, il secondo è **center**)
 - ▶ come distanza dall'angolo in alto a sinistra dell'elemento contenitore
 - ▶ tramite coppia di valori percentuali, dove 0% 0% corrisponde all'angolo in alto a sinistra e 100% 100% all'angolo in basso a destra
- ▶ Con CSS3 anche background multipli:
[Esempi CSS\Colori\immagini_sfondo_2.html](#)

Immagini in background (3)

- ▶ **background-attachment** permette di decidere se un'immagine di background deve scrollare assieme al contenuto dell'elemento in cui è inserita (valore **scroll**, default) o se deve rimanere fissa
- ▶ **background-size** permette di definire la dimensione dell'immagine di sfondo:
 - ▶ **auto** (default) usa la dimensione originale dell'immagine
 - ▶ **cover** occupa tutto il container facendo stretch e tagli (mantenendo proporzioni)
 - ▶ **contain** ridimensiona per far stare l'immagine intera (mantenendo proporzioni)
 - ▶ **un valore** di lunghezza / percentuale (rispetto al padre) imposta la larghezza (altezza è auto)
 - ▶ **due valori** di lunghezza / percentuale impostano la larghezza e l'altezza
- ▶ [Esempi CSS\Colori\background-attachment-and-size.html](#)

Esercizio: pulsante con sprite

- ▶ Data:
 - ▶ l'immagine «button-sprite.jpg» nella cartella «images»
 - ▶ il file HTML «image-rollovers-and-sprites.html»
- ▶ Creare:
 - ▶ il file «image-rollovers-and-sprites.css» da inserire nella cartella «css»
- ▶ Per fare in modo che:
 - ▶ i pulsanti con id «add-to-basket» e «framing-options» utilizzino le diverse immagini in «button-sprite.jpg» quando l'utente è sopra i pulsanti e quando li clicca

Gradienti

- ▶ I gradienti lineari variano da un lato ad un altro (vedi esempio)
 - ▶ **linear-gradient()** ha per argomenti la direzione in cui si sviluppa e i colori attraverso i quali deve passare (minimo due)
 - ▶ La direzione può essere specificata tramite valore angolare o parola chiave (0deg = dal basso verso l'alto, 90deg = da sinistra verso destra,...; le parole chiave specificano una direzione in incrementi di 90 gradi, *to top*, *to right*,...)
 - ▶ I colori sono seguiti da un valore percentuale che indica in quale punto nella direzione indicata deve essere visualizzato quel colore
- ▶ I gradienti radiali si espandono da un punto verso l'esterno (vedi esempio)
 - ▶ **radial-gradient()** ha per argomenti la forma ellipse o circle, la dimensione (lunghezza del raggio o parola chiave che indica il lato o angolo in cui deve fermarsi), **at** posizione del centro (come background-position), i colori
- ▶ Per semplificare la creazione dei gradienti è possibile utilizzare generatori online come <http://www.colorzilla.com/gradient-editor/>

Famiglie di font

- ▶ Ripasso famiglie di font:
 - ▶ Serif (grazie) -> meglio per la stampa, es. Georgia, Times New Roman
 - ▶ Sans-Serif -> meglio per lo schermo, es. Arial, Helvetica
 - ▶ Monospace -> meglio per il codice, es. Courier
 - ▶ Font particolari con stili in corsivo o più fantasiosi, es. Comic



Proprietà font-family

- ▶ Con font-family, il browser può mostrare un font scelto
- ▶ È possibile specificare più di un font in ordine di preferenza (**font stack**)
- ▶ È bene chiudere lo stack con un fallback della famiglia

```
body {  
  font-family: Georgia, Times, serif;}  
h1, h2 {  
  font-family: Arial, Verdana, sans-serif;}  
.credits {  
  font-family: "Courier New", Courier,  
    monospace;}
```

- ▶ Pro:
 - ▶ Veloce, perché non deve scaricare nulla
 - ▶ Nessun problema di licenze, perché non si distribuisce nulla
- ▶ Contro:
 - ▶ Se l'utente non ha installato il font, viene usato uno diverso
 - ▶ Font su PC diversi da font su Mac
 - ▶ Conviene limitarsi a font comuni che hanno tutti

Usare altri font

- ▶ Con @font-face si possono far scaricare all'utente altri font
- ▶ Pro:
 - ▶ Si possono usare più font
 - ▶ Esistono siti con font gratuiti come www.fontsquirrel.com e www.dafont.com
- ▶ Contro:
 - ▶ Tempo di scaricamento
 - ▶ Problemi di licenze (uso personale)
 - ▶ Costo dei servizi
 - ▶ Compatibilità dei browser con i diversi formati

```
@font-face {  
  font-family: 'ChunkFiveRegular';  
  src: url('fonts/chunkfive.eot');  
h1, h2 {  
  font-family: ChunkFiveRegular, Georgia, serif;}  
}
```

- ▶ Altre possibilità
 - ▶ Usare immagini (sconsigliato)
 - ▶ Usare servizi come Google Fonts <https://fonts.google.com/>
 - ▶ Esempio: [Esempi CSS\Testi\testo.html](#)

Proprietà del font

- ▶ Scelto un font possiamo cambiare:
 - ▶ **font-weight**, cambia enfasi e contrasto (**lighter**, **normal**, **bold**, **bolder**)
 - ▶ **font-style**, mette in obliquo e in corsivo (**normal**, **oblique**, **italic**)
 - ▶ **font-stretch**, comprime o espande le lettere (**condensed**, **normal**, **expanded**)

WEIGHT

Light

Medium

Bold

Black

STYLE

Normal

Italic

Oblique

STRETCH

Condensed

Regular

Extended

Dimensione del font e della linea

- ▶ La dimensione del font si imposta con **font-size**
- ▶ È possibile utilizzare diverse unità di misura
 - ▶ Pixel
 - ▶ Percentuale (rispetto al font-size del contenitore oppure nel body al font-size standard di 16px)
 - ▶ Em (larghezza della lettera m)
 - ▶ Point (pt, 1/72 di pollice)
- ▶ L'altezza della linea si imposta con **line-height**

Anticipazione:
Prossimamente vedremo
anche vh, vw

```
body {  
  font-family: Arial, Verdana, sans-serif;  
  font-size: 12px;}  
h1 {  
  font-size: 200%;}  
h2 {  
  font-size: 1.3em;}
```

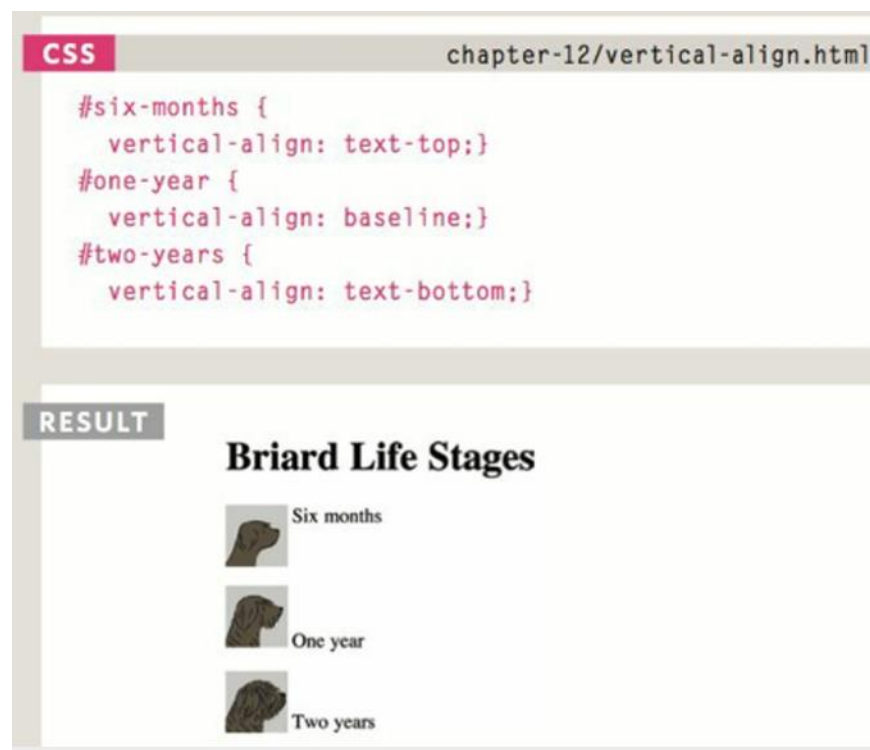
```
p {  
  line-height: 1.4em;}
```

Modifiche al testo

- ▶ **text-transform** permette di trasformare il testo in:
 - ▶ maiuscolo (**uppercase**)
 - ▶ minuscolo (**lowercase**)
- ▶ O rendere maiuscola la prima lettera di ogni parola (**capitalize**)
- ▶ C'è anche **font-variant: small-caps** per il maiuscoletto
- ▶ **letter-spacing** e **word-spacing** cambiano lo spazio tra lettere e parole (in em aggiuntivi)
- ▶ **text-decoration** permette di
 - ▶ Sottolineare (**underline**)
 - ▶ Aggiungere una linea sopra (**overline**)
 - ▶ Barrare (**line-through**)
 - ▶ Far lampeggiare (**blink**)
 - ▶ Rimuovere decorazioni (**none**)
- ▶ **text-shadow** applica l'ombra alla distanza, con il blur e il colore specificati
- ▶ **word-wrap** (break-word o normal) permette di far andare a capo o meno le parole che uscirebbero dal box

Allineamento del testo

- ▶ **text-align** imposta l'allineamento orizzontale a sinistra (**left**), destra (**right**), centrato (**center**) o giustificato (**justify**)
- ▶ **vertical-align** imposta l'allineamento verticale di un elemento inline rispetto al testo
 - ▶ Oltre ai valori in figura può prenderne altri (es., **middle**), una lunghezza, una percentuale del `line-height`,...
- ▶ **text-indent** sposta la prima riga di un testo



Gestione delle liste

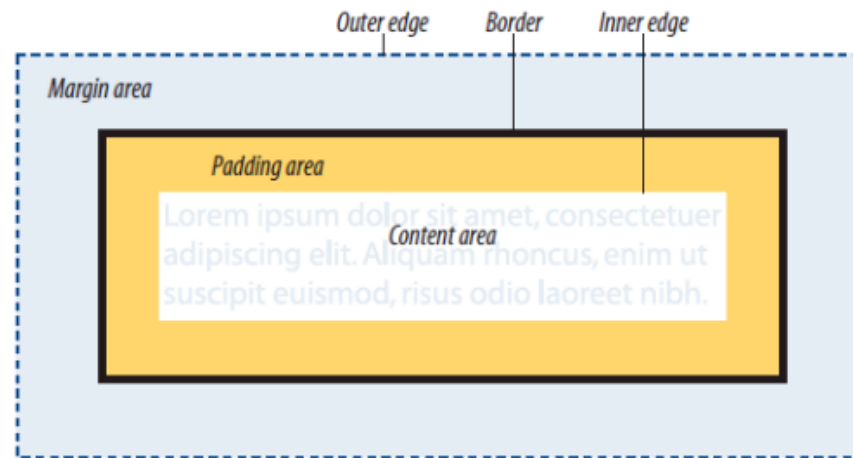
- ▶ Queste proprietà si applicano a ``, `` e `` o ad elementi con proprietà **display** impostata a **list-item** (per applicare stile lista anche ai paragrafi)
- ▶ **list-style-type** permette di scegliere il simbolo che precede gli elementi della lista
- ▶ **list-style-position** specifica se il simbolo deve far parte del contenuto di una lista (valore **inside**) oppure no (valore **outside**)

Keyword	System
decimal	1, 2, 3, 4, 5...
decimal-leading-zero	01, 02, 03, 04, 05...
lower-alpha	a, b, c, d, e...
upper-alpha	A, B, C, D, E...
lower-latin	a, b, c, d, e... (same as lower-alpha)
upper-latin	A, B, C, D, E... (same as upper-alpha)
lower-roman	i, ii, iii, iv, v...
upper-roman	I, II, III, IV, V...
lower-greek	α , β , γ , δ , ϵ ...

- ▶ **list-style-image** permette di specificare un'immagine da impiegare come simbolo (il valore è l'URL dell'immagine)

Box Model

- ▶ Ripasso:
 - ▶ immaginiamo **ogni** elemento HTML dentro **un box**
 - ▶ il box è inizialmente invisibile, mentre il suo contenuto è visibile
- ▶ Con CSS, possiamo controllare:
 - ▶ Dimensione dei box
 - ▶ Bordi attorno ai box
 - ▶ Margini e padding («imbottitura»)
 - ▶ Visibilità
 - ▶ Posizione

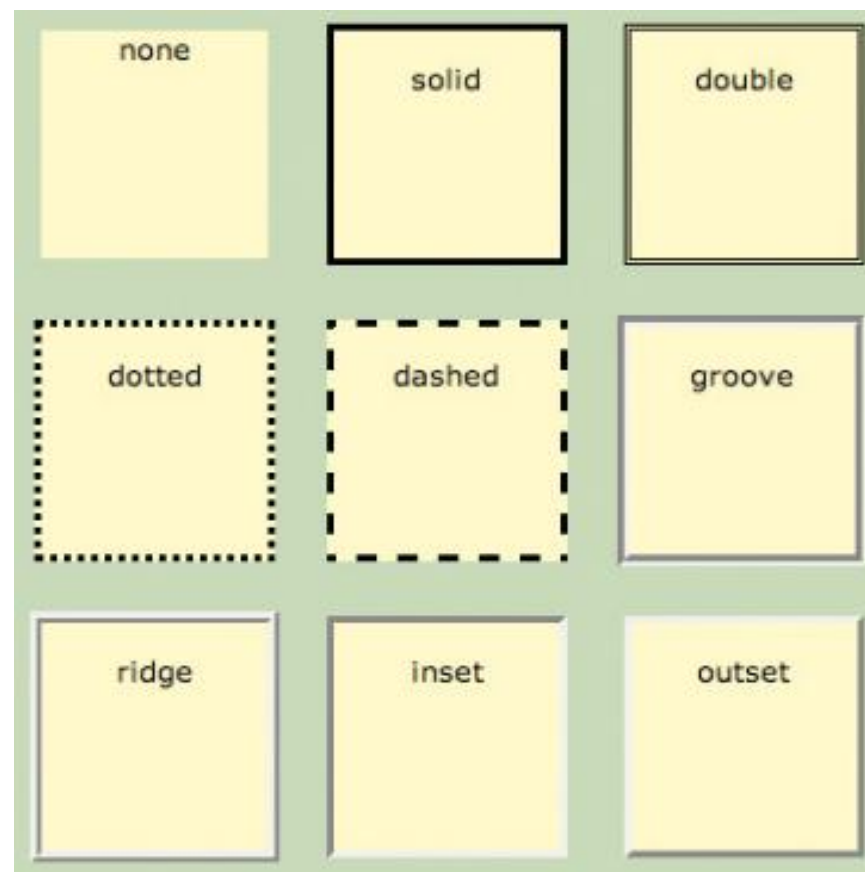


Padding

- ▶ Per aggiungere spazio di padding tra contenuto e bordo, si utilizza
 - ▶ `padding` (con 4, 3, 2 o 1 valore)
 - ▶ `padding-top`
 - ▶ `padding-right`
 - ▶ `padding-bottom`
 - ▶ `padding-left`
- ▶ I valori sono misure di lunghezza o valori percentuali (riferite alla larghezza dell'elemento genitore)
- ▶ Esempio: [Esempi CSS\Box\padding.html](#)

Bordi (style)

- ▶ Il bordo è l'area (opzionale) tra padding e margine
- ▶ Viene utilizzato per rendere visibile l'area costituita da contenuto e padding
- ▶ Se non viene definito lo stile, non viene visualizzato
- ▶ È possibile impostare lo stile di un bordo con **border-style** o **border-top-style**,...
- ▶ I possibili valori sono le parole chiave in figura
- ▶ Esempio:
<Esempi CSS\Box\bordi.html>

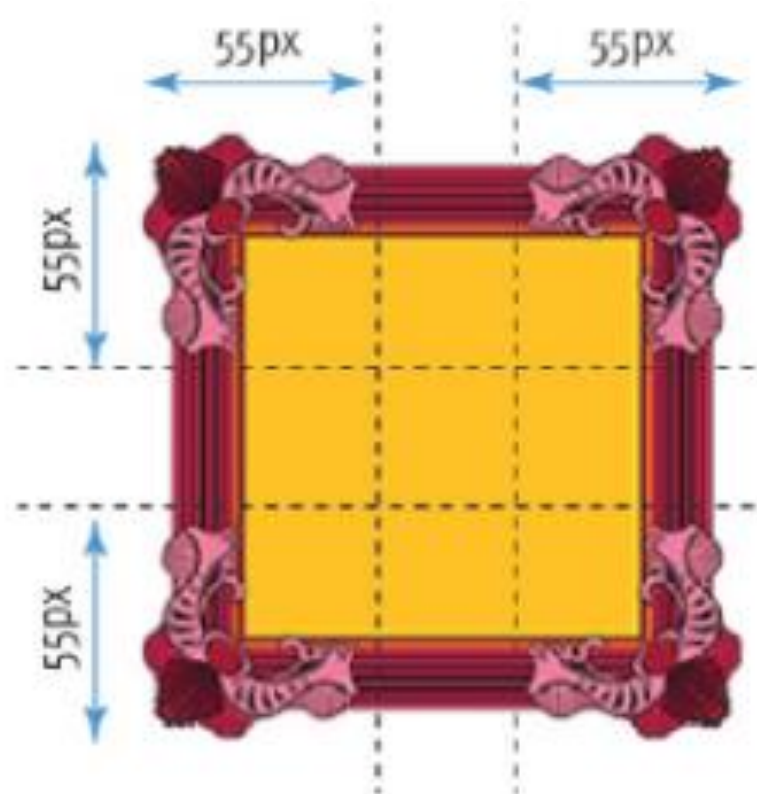


Bordi (altre proprietà)

- ▶ **border-width** permette di definire lo spessore con le unità di misura di lunghezza o le parole chiave **thin**, **medium** e **thick**
- ▶ **border-color** permette di specificare il colore (andando a sovrascrivere il foreground definito con color) o la parola chiave **transparent** (rende il bordo trasparente, ma mantiene le dimensioni)
- ▶ **border-radius** permette di arrotondare gli angoli specificando i valori con le unità di lunghezza o percentuale (riferita alla larghezza del box)
- ▶ Esempi:
 - ▶ [Esempi CSS\Box\border-width.html](#)
 - ▶ [Esempi CSS\Box\border-color-and-radius.html](#)

Bordi con immagini

- ▶ **border-image** permette di definire un bordo complesso basato su un'immagine
- ▶ Il primo valore specifica l'immagine da utilizzare
- ▶ Il secondo permette di dividere l'immagine in 9 parti in termini di distanza dal bordo
- ▶ L'ultimo determina come viene gestita la parte centrale:
 - ▶ stretch consente di allungarla fino a riempire
 - ▶ repeat permette di ripeterla
 - ▶ round e space permettono ripetizioni che evitino pezzi parziali
- ▶ [Esempi CSS\Box\border-image.html](#)



Margini

- ▶ Evitano che gli elementi finiscano uno attaccato all'altro
- ▶ Di default, il browser aggiunge margini attorno ad elementi di tipo blocco
- ▶ **margin**, **margin-top**, **margin-right**,... impostano il margine nelle varie unità, percentuali (rispetto al padre), o con **auto** per riempire lo spazio a disposizione (e centrare come in [Esempi CSS\Box\centering-content.html](#))
- ▶ Margini adiacenti verticali (top e bottom) vengono collassati (si usa solo il più grande) ad eccezione degli elementi floating o posizionati in modo assoluto (ne parleremo più avanti)
- ▶ Margini destri e sinistri non vengono mai collassati
- ▶ Per gli elementi inline di tipo testo, i margini vengono rispettati a destra e sinistra, ma i margini inferiore e superiore no (non alterano l'altezza delle righe)
- ▶ Per gli elementi inline rimpiazzati (es. immagini), vengono considerati tutti
- ▶ È possibile specificare margini negativi (può essere utile per gestire il layout)
- ▶ Esempio: [Esempi CSS\Box\margin.html](#)

Dimensione dei box

- ▶ Di default, i box sono grandi
 - ▶ grandi quanto basta per il contenuto, nel caso degli inline
 - ▶ larghi quanto la pagina o contenitore e alti quanto basta, nel caso dei block
- ▶ Per cambiare larghezza e altezza usiamo le proprietà:
 - ▶ width
 - ▶ height
- ▶ Possiamo usare le diverse unità di misura: pixel, percentuale, em,... oppure auto

```
div {  
  height: 300px;  
  width: 400px;  
  background-color: #ee3e80;}  
p {  
  height: 75%;  
  width: 75%;  
  background-color: #e1ddda;}
```

- ▶ box-sizing specifica se le misure si applicano al contenuto (content-box) o al bordo (border-box)

Limiti per design fluidi

- ▶ Nel caso di design fluidi che si espandono in base alla finestra, è possibile impostare dei **limiti** con:
 - ▶ `min-width`
 - ▶ `max-width`
 - ▶ `min-height`
 - ▶ `max-height`
- ▶ Può essere molto utile, ad esempio nelle **tabelle**
- ▶ Con **overflow** (**visible**, **hidden**, **scroll** o **auto**) diciamo cosa succede quando il contenuto non sta nei limiti

```
HTML chapter-13/min-width-max-width.html
<tr>
  <td></td>
  <td class="description">The Rhodes piano is an
    electro-mechanical piano, invented by Harold
    Rhodes during the fifties and later
    manufactured in a number of models ...</td>
  <td>$1400</td>
</tr>

CSS
td.description {
  min-width: 450px;
  max-width: 650px;
  text-align: left;
  padding: 5px;
  margin: 0px;}
```

Ombreggiatura

- ▶ box-shadow (CSS3) permette di aggiungere un'ombreggiatura
- ▶ È l'equivalente di text-shadow per il testo
- ▶ L'ombra viene aggiunta alla parte visibile del box, cioè (di default) all'esterno del bordo
- ▶ [Esempi CSS\Box\box-shadow.html](#)

Visualizzazione

- ▶ **display** cambia la visualizzazione dell'elemento in inline, block, list-item, table, flex (che useremo per i layout flessibili), none,...
- ▶ non modifica il tipo HTML (es. inserire `<p>` in un inline non è valido anche se `<p>` è visualizzato inline)
- ▶ Esempio: è possibile trasformare elementi `` in elementi inline per creare menu orizzontali ([Esempi CSS\Box\display-menu.html](#))
- ▶ **display: none** eliminare completamente un elemento dalla visualizzazione, mentre **visibility: hidden** nasconde il contenuto, ma lascia lo spazio occupato

Stile delle tabelle

- ▶ Si possono utilizzare le proprietà che abbiamo visto in precedenza per il normale contenuto e alcune proprietà specifiche delle tabelle
- ▶ **border-collapse** determina come mostrare i bordi tra celle adiacenti
 - ▶ **separate**: ogni cella ha un proprio bordo ed è possibile definire la distanza tra i bordi di celle adiacenti utilizzando **border-spacing** (due valori di lunghezza, uno per lo spazio tra colonne ed uno per lo spazio tra righe, o un solo valore comune)
 - ▶ **collapse**: viene visualizzato un solo bordo tra le celle
- ▶ Nel caso di bordi separati, è anche possibile utilizzare **empty-cells** per decidere se visualizzare o meno i bordi delle celle vuote (**show** o **hide**)
- ▶ Esempio: [Esempi CSS\Box\tabella.html](#)

Posizionamento

- ▶ Finora abbiamo visto solo esempi con il posizionamento standard
- ▶ Flusso normale:
 - ▶ Elementi di tipo **blocco** visualizzati uno di seguito all'altro verticalmente
 - ▶ Elementi di tipo **inline** posizionati all'interno del flusso del testo
- ▶ CSS consente di controllare direttamente il posizionamento degli elementi tramite i metodi di
 - ▶ **Floating**
 - ▶ **Positioning**

Floating

- ▶ Si utilizza la proprietà **float**, che si può applicare a tutti gli elementi
- ▶ Permette di muovere un elemento all'estrema destra (valore **right**) o all'estrema sinistra (valore **left**) dell'area di contenuto del padre
- ▶ La proprietà si applica a tutto l'elemento inclusi padding, margini e bordi
- ▶ L'elemento viene inserito dentro il padding del padre, ma può uscire
- ▶ Anche se non ha più il comportamento tradizionale, l'elemento float continua ad influenzare il contenuto circostante: il contenuto che lo segue all'interno dello stesso padre si disporrà attorno all'elemento, nello spazio disponibile
- ▶ Questa proprietà non viene ereditata
- ▶ Esempio: [Esempi CSS\Posizionamento\floating.html](#)

Floating di inline

- ▶ Esempio: [Esempi CSS\Posizionamento\floating-inline.html](#)
- ▶ È preferibile specificare la proprietà **width** per un testo inline floating poiché le dimensioni del box dell'elemento potrebbero essere diverse da browser a browser
- ▶ Gli elementi inline floating si comportano come elementi di tipo blocco e quindi i margini vengono visualizzati su tutti i lati (non solo destra e sinistra)
- ▶ I margini superiore e inferiore degli elementi floating non vengono uniti

Floating di block

- ▶ Esempio: [Esempi CSS\Posizionamento\floating-block.html](#)
- ▶ Se non si imposta la **width** dell'elemento, questo occuperà tutto lo spazio che occupa normalmente
- ▶ Un elemento floating di tipo blocco starà sempre al di sotto degli elementi blocco che lo precedono
- ▶ Per posizionare un elemento floating nell'angolo in alto a sinistra della finestra, è necessario che l'elemento sia il primo nel codice html (o usare il posizionamento assoluto, che vedremo)
- ▶ Per allineare i margini superiori dell'elemento floating e dell'elemento seguente, è necessario impostare il margine superiore dell'elemento floating a 0 (altrimenti usa il default del browser)

Annulare il floating

- ▶ Per ripristinare il normale comportamento degli elementi dopo un elemento floating, è necessario utilizzare la proprietà **clear**
- ▶ **clear** si applica solo agli elementi di tipo blocco e deve essere applicata all'elemento che si vuole venga posizionato **sotto** all'elemento floating (non all'elemento floating)
 - ▶ Il valore left posiziona un elemento sotto qualsiasi elemento con proprietà “float” impostata a left
 - ▶ Il valore right posiziona un elemento sotto qualsiasi elemento con proprietà “float” impostata a right
 - ▶ Il valore both posiziona un elemento sotto qualsiasi elemento floating
- ▶ Esempio: [Esempi CSS\Posizionamento\floating-clear.html](#)

Floating multipli

- ▶ È possibile inserire più elementi floating nella stessa pagina o elemento
- ▶ Un sistema di regole fa sì che gli elementi floating non si sovrappongano (posiziona gli elementi più a destra o sinistra possibile e più in alto possibile)
- ▶ Esempio: [Esempi CSS\Posizionamento\floating-multipli.html](#)
- ▶ Se vengono resi floating tutti gli elementi di un certo contenitore, quel contenitore «scompare»
- ▶ Soluzioni: rendere il contenitore floating e impostarne le dimensioni al 100% o impostare la proprietà **overflow** del contenitore ad auto o hidden
- ▶ Esempi:
 - ▶ [Esempi CSS\Posizionamento\floating-contenitore.html](#)
 - ▶ [Esempi CSS\Posizionamento\floating-menu.html](#)

Floating per layout a colonne

- ▶ float può essere sfruttata per organizzare il layout di un sito in colonne
- ▶ Per creare un layout a due colonne, si possono utilizzare le seguenti soluzioni:
 - ▶ Rendere floating a sinistra un primo <div> e aggiungere un margine sinistro sufficiente ad un secondo <div>
 - ▶ Rendere floating a destra o a sinistra due <div>
 - ▶ Rendere floating a sinistra un div e a destra un altro
- ▶ Nella creazione di layout di questo tipo bisogna impostare bene le dimensioni (in larghezza) degli elementi floating (inclusi padding, bordi e margini)
- ▶ Se la dimensione totale è superiore allo spazio disponibile, gli elementi floating in eccesso vengono riposizionati più in basso

Positioning

- ▶ Consente di posizionare precisamente gli elementi sulla pagina
- ▶ **position** permette di indicare quale metodo utilizzare:
 - ▶ **static** è il default visto finora
 - ▶ **relative** è un posizionamento relativo rispetto al normale flusso
 - ▶ **absolute** posiziona in modo assoluto rispetto al primo antenato non statico
 - ▶ **fixed** posiziona in modo fisso rispetto alla finestra del browser e la posizione viene mantenuta anche durante lo scrolling
- ▶ **top, right, bottom, left** permettono di specificare l'offset (lunghezza o percentuale) dell'elemento rispetto al contenitore/antenato/finestra (valori negativi definiscono un offset nella direzione opposta)

Posizionamento relativo

- ▶ Il posizionamento relativo (**relative**) muove un elemento rispetto al posto che questo avrebbe normalmente nel flusso degli elementi
- ▶ Esempio: [Esempi CSS\Posizionamento\posizionamento-relativo.html](#)
- ▶ Lo spazio che l'elemento avrebbe occupato normalmente viene mantenuto
- ▶ L'elemento nella nuova posizione può andare a sovrapporsi ad altri elementi

Posizionamento assoluto

- ▶ Nel posizionamento assoluto (**absolute**) lo spazio nel flusso non viene mantenuto e non influisce sul posizionamento di altri elementi
- ▶ Il posizionamento assoluto viene effettuato rispetto al più vicino antenato che utilizza position con valore diverso da static
- ▶ Se non c'è nessun antenato di questo tipo, allora il posizionamento è assoluto rispetto alla radice <html>, ovvero alla finestra del browser
- ▶ Quando l'antenato è block, il posizionamento viene calcolato rispetto al limite del padding, mentre quando è inline, rispetto all'area del contenuto
- ▶ Un inline posizionato in modo assoluto si comporta come un block
- ▶ Non combinare dimensioni e posizionamento assoluto su ogni lato
- ▶ Esempi: [Esempi CSS\Posizionamento\posizionamento-assoluto.html](#)

Sovrapposizioni

- ▶ Gli elementi posizionati possono sovrapporsi tra di loro
- ▶ **z-index** permette di modificare l'ordine con cui tali elementi vengono visualizzati sullo schermo
- ▶ Di default, gli elementi appaiono nell'ordine in cui vengono incontrati nel documento html
- ▶ Il valore è un numero (positivo o negativo) che determina l'ordine: gli elementi vengono visualizzati a partire dai valori più bassi (i valori utilizzati non sono importanti, conta solo l'ordine relativo)
- ▶ Esempio: [Esempi CSS\Posizionamento\z-index.html](#)

Posizionamento fisso

- ▶ Il posizionamento fisso (**fixed**) si comporta in modo simile a quello assoluto
- ▶ La differenza è che lo spostamento degli elementi è sempre relativo alla finestra del browser
- ▶ Gli elementi posizionati in questo modo restano fissi nella loro posizione sullo schermo anche quando l'utente scrolla la pagina
- ▶ Può essere utilizzato per mantenere un menu sempre visibile sullo schermo
- ▶ Esempio: [Esempi CSS\Posizionamento\posizionamento-fisso.html](#)

Strategie di layout

Ora anche layout
FlexBox

- ▶ Diverse strategie possibili per il layout:
- ▶ Layout **fisso**: mantiene le proprie dimensioni indipendentemente dalla dimensione della finestra del browser o del testo
- ▶ Layout **fluid** o **liquido**: le dimensioni del layout cambiano in modo proporzionale alle dimensioni della finestra
- ▶ Layout **elastico**: le dimensioni del layout cambiano in modo proporzionale alla dimensione del testo
- ▶ Layout **ibrido**: combina aree fisse e aree le cui dimensioni variano dinamicamente

Layout fissi

- ▶ I designer decidono la dimensione delle pagine e le relazioni tra elementi delle pagine, allineamenti e lunghezza delle linee
- ▶ Vengono creati impostando in **pixel** la larghezza delle pagine (una dimensione comune è/era **960px**) e dei loro componenti (colonne, margini,...)
- ▶ Tipicamente, il contenuto dell'intera pagina viene inserito all'interno di un elemento `<div>` posizionato al centro (`margin: 0 auto`) della pagina in modo che eventuale spazio extra si disponga equamente
- ▶ Gli svantaggi sono dovuti al fatto che non si adatta alla dimensione della finestra del browser che può quindi diventare più piccola o molto più grande del contenuto da visualizzare
- ▶ Pensato quando Internet era visualizzato su schermi 1024x768 o simile
- ▶ Esempio: <Esempi CSS/Layout/layout-fisso.html>

Layout fluidi

- ▶ Nei layout fluidi, la dimensione delle pagine e delle colonne varia **dinamicamente** in funzione dello spazio
- ▶ Vengono creati impostando la larghezza delle pagine e degli elementi in percentuale (o senza specificare **width**, per cui diventa **auto**)
- ▶ Il principale svantaggio è che possono portare a righe di testo ed altri elementi molto lunghi o corti, pregiudicando la leggibilità
- ▶ Per mitigare questo problema si può far uso di **min-width** e **max-width**
- ▶ Esempio: [Esempi CSS/Layout/layout-fluido.html](#)

Layout elastici

- ▶ Nei layout elastici, la dimensione delle pagine e degli elementi si adatta alla dimensione del testo
- ▶ Questo permette di avere righe sempre della stessa lunghezza (in termini di numero di parole o caratteri)
- ▶ I layout elastici vengono creati impostando le dimensioni degli elementi in **em**
- ▶ Questo tipo di layout si applica meno bene a contenuti non testuali
- ▶ Esempio: [Esempi CSS/Layout/layout-elastici.html](#)

Layout multicolonna

- ▶ Si possono realizzare con **float** e con **position**
- ▶ Alcuni esempi di layout a due e a tre colonne:
 - ▶ [Esempi CSS\Layout\2col-float-fluido.html](#)
 - ▶ [Esempi CSS\Layout\3col-position-fluido.html](#)
- ▶ Altri esempi negli esercizi

CSS Reset

- ▶ I browser hanno un loro foglio di stile che viene applicato se alcune proprietà non sono definite
- ▶ Tale foglio di stile può essere diverso da browser a browser, portando a risultati incoerenti
- ▶ Molto spesso i designer utilizzano il CSS Reset: un insieme di regole che sovrascrive tutti gli stili del browser in modo che non interferiscano con le regole definite esplicitamente
- ▶ Una delle implementazioni più diffuse è stata scritta da Eric Meyer:
<http://meyerweb.com/eric/tools/css/reset/>

Responsive design

- ▶ Tecnica che adatta il layout in base alla dimensione dello schermo
- ▶ È utile per realizzare layout diversi per dispositivi mobili e desktop
- ▶ Tre aspetti principali:
 - ▶ dimensione del viewport
 - ▶ layout fluido (e FlexBox)
 - ▶ media query

Dimensione del viewport

- ▶ Il **viewport** è l'area visibile dall'utente (dimensione finestra)
- ▶ Con l'elemento **<meta>** si controlla la dimensione iniziale del viewport
- ▶ Il **<meta>** nell'esempio informa il browser di definire la dimensione del viewport uguale alla dimensione dello schermo del dispositivo
- ▶ L'attributo **initial-scale** definisce il livello di zoom iniziale (1=100%)
- ▶ Esempio: [Esempi CSS\Responsive design\layout-fisso-meta.html](#)

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport"
content="width=device-width, initial-
scale=1">
    <title>Tecnologie web</title>
  </head>
  <body>
  </body>
</html>
```

- ▶ Nuove unità di misura: **vw**, **vh**, **vmin**, **vmax** (1% della corrispondente dimensione del viewport)

Layout fluido

- ▶ I layout fissi 960px al normale livello di zoom posso richiedere lo scroll orizzontale su dispositivi mobili
- ▶ Non è possibile creare un layout differente per ciascuna possibile misura di schermo
- ▶ Si possono creare due o tre layout per le classi di dispositivi principali (smartphone, tablet, desktop)
- ▶ Si sfruttano le proprietà dei layout fluidi per le dimensioni intermedie
- ▶ Esempio: Esempi CSS\Responsive design\layout-fluido-meta.html
- ▶ Impostare la proprietà **max-width** dell'elemento `` al valore 100% affinché venga scalata per occupare solo lo spazio disponibile nel contenitore quando diventa più piccolo dell'immagine (non impostare width ed height)

Media query (1)

- ▶ Sono direttive CSS per specificare regole da applicare solo in particolari condizioni relative al media
- ▶ Iniziano con @media seguita dal tipo di media e da zero o più espressioni (in and) con le condizioni di (non) validità delle caratteristiche del media (ogni espressione va inserita tra parentesi tonde)
- ▶ Le media query possono essere specificate nel foglio di stile o usando l'attributo **media** di <link> per caricare i fogli di stile in modo condizionale

```
@media screen and (min-width: 480px) {  
    /* regole di stile */  
}
```

```
<head>  
    <link rel="stylesheet" href="stile.css">  
    <link rel="stylesheet" href="stile2.css"  
    media="screen and (min-width:780px)">  
</head>
```

Media query (2)

- ▶ I tipi di media specificabili sono:

- ▶ all (tutti i dispositivi)
- ▶ screen (schermo di computer, tablet, smartphone)
- ▶ print (pagina stampata)
- ▶ speech (sintesi vocale)
- ▶ ...

- ▶ Alcune caratteristiche più comuni che si possono testare:

- ▶ min-width
- ▶ max-width
- ▶ min-height
- ▶ max-height
- ▶ orientation
- ▶ aspect-ratio
- ▶ ...

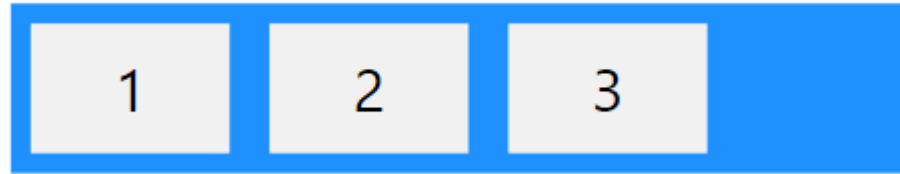
Mobile first

- ▶ Una strategia molto diffusa oggi è quella **mobile first**:
 - ▶ prima vengono scritte le regole di stile per i dispositivi mobili più semplici
 - ▶ poi le media query per i dispositivi con le caratteristiche più avanzate
 - ▶ si usa min-width per specificare quali proprietà si applicheranno solo ai viewport con risoluzione maggiore
- ▶ Esempio: [Esempi CSS\Responsive design\mobile-first.html](#)
- ▶ In alternativa c'è anche approccio desktop down (usando max-width)
- ▶ Di solito si usano dei breakpoint: 320, 768, 1024

FlexBox

- ▶ Abbiamo visto vari tipi di display:

- ▶ inline
- ▶ block
- ▶ list-item
- ▶ inline-block
- ▶ ...



- ▶ Ora vediamo **display: flex** che serve a definire un **contenitore flessibile** (come l'elemento con sfondo blu in figura)
- ▶ I figli diretti di un contenitore flex diventano automaticamente **item flex** (come gli elementi grigi numerati 1, 2 e 3 in figura)

Proprietà dei contenitori flex (1)

- ▶ **flex-direction** determina la direzione in cui disporre gli elementi:
 - ▶ **column** dispone in verticale
 - ▶ **column-reverse** dispone in verticale, ma da basso in alto
 - ▶ **row** dispone in orizzontale
 - ▶ **row-reverse** dispone in orizzontale, ma da destra a sinistra
- ▶ **flex-wrap** determina se gli elementi possono andare a capo:
 - ▶ **wrap** fa andare a capo
 - ▶ **no-wrap** impedisce che vadano a capo (si schiacciano)
 - ▶ **wrap-reverse** fa andare a capo in ordine inverso
- ▶ **flex-flow** permette di specificare sia direzione che a capo
- ▶ Esempio: [Esempi CSS\Responsive design\flex-direction-wrap.html](#)

Proprietà dei contenitori flex (2)

- ▶ **justify-content** serve per la disposizione sulla direzione principale:
 - ▶ **center** dispone al centro
 - ▶ **flex-start** dispone all'inizio del contenitore
 - ▶ **flex-end** dispone alla fine del contenitore
 - ▶ **space-around** mette spazio prima, dopo e tra gli elementi
 - ▶ **space-between** mette spazio tra gli elementi
- ▶ Esempio: [Esempi CSS\Responsive design\flex-justify.html](#)
- ▶ **align-items** serve per allineare rispetto alla direzione secondaria:
 - ▶ **center** allinea al centro
 - ▶ **flex-start** allinea all'inizio
 - ▶ **flex-end** allinea alla fine
 - ▶ **stretch** riempie il contenitore
 - ▶ **baseline** allinea rispetto alla baseline
- ▶ Esempio: [Esempi CSS\Responsive design\flex-align-items.html](#)
- ▶ C'è anche **align-content**
- ▶ Esempio: [Esempi CSS\Responsive design\flex-align-items-content.html](#)

Proprietà dei flex item

- ▶ **order** imposta l'ordine in cui l'elemento comparirà
- ▶ **flex-grow** imposta il fattore di crescita rispetto agli altri item (0 non cresce)
- ▶ **flex-shrink** imposta il fattore di compressione rispetto agli altri item (0 non si comprime)
- ▶ **flex-basis** imposta la lunghezza iniziale
- ▶ **align-self** imposta un comportamento di allineamento specifico per l'item (sovrascrive align-items del contenitore)
- ▶ Esempio: <Esempi CSS\Responsive design\flex-item.html>

Flex e media query

- ▶ Utilizzando la proprietà **flex** è possibile specificare assieme flex-grow, flex-shrink e flex-basis
- ▶ È possibile usare flex anche per specificare una percentuale di spazio del contenitore da occupare, ad esempio flex: 50%
- ▶ Combinando flex e media query si può cambiare direzione da riga a colonna (o viceversa) e cambiare le percentuali dei flex su dispositivi a diverse risoluzioni
- ▶ Esempio: <Esempi CSS\Responsive design\flex-e-media-query.html>