ACCADEMIA DI BELLE ARTI
ACADEMY OF FINE ARTS AND MEDIA

Web Design

Matteo Baccan

CSS

CSS

Cascading Style Sheets : in italiano fogli di stile a cascata

È un linguaggio usato per definire la formattazione di documenti
Descrive lo stile degli elementi di pagina
La sua interpretazione necessita di un browser

CSS – strumenti

Lo strumento che useremo durante il corso è

https://codepen.io

CodePen is a social development environment. At its heart, it allows you to write code in the browser, and see the results of it as you build. A useful and liberating online code editor for developers of any skill, and particularly empowering for people learning to code. We focus primarily on front-end languages like HTML, CSS, JavaScript, and preprocessing syntaxes that turn into those things.

Iscrivetevi e seguite il profilo creato apposta per il corso

https://codepen.io/matteobaccan

CSS – strumenti

Editor

Codepen.io
Notepad
Notepad++
VisualStudio Code

Va bene qualsiasi editor, non visuale, meglio se con syntax highlighter e code completion

Le slide e i sorgenti del corso, liberamente ispirati a https://www.w3schools.com e costantemente aggiornati, sono disponibili a questo indirizzo

https://github.com/matteobaccan/CorsoCSS

CSS – esempio

```
body {
  background-color: red;
}

h1 {
  color: black;
  text-align: center;
}

p {
  font-family: courier;
  font-size: 24px;
}
```

CSS

Cos'è il CSS?

CSS è acronimo di **Cascading Style Sheets**, sono fogli che vengono utilizzati per formattare le pagine web.

Con i CSS è possibile controllare il colore, il carattere, la dimensione del testo, la spaziatura tra gli elementi, il modo in cui gli elementi sono posizionati e disposti, quali immagini di sfondo o colori di sfondo devono essere utilizzati, o le diverse visualizzazioni in base alle dimensioni dello schermo

Da notare che **cascading** identifica il fatto che uno stile applicato a un elemento padre si applicherà anche a tutti gli elementi figli all'interno dell'elemento padre

CSS

I CSS possono essere aggiunti ai documenti HTML in 3 modi:

Inline - utilizzando l'attributo style all'interno degli elementi HTML
Interno - utilizzando un elemento <style> nella sezione <head>
Esterno: utilizzando un elemento <link> per collegarsi a un file CSS esterno

CSS Inline

```
<h1 style="color:blue;">Una intestazione blue</h1>

<p style="color:red;">Un paragrafo rosso</p>
```

CSS Interno

```
<!DOCTYPE html>
<html>
 <head>
  <style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
  </style>
 </head>
 <body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
 </body>
</html>
```

CSS Esterno

```
<!DOCTYPE html>
 <html>

  <head>
   <link rel="stylesheet" href="styles.css">
  </head>

 <body>
  <h1>Intestazione</h1>
  <p>Paragrafo</p>
 </body>

</html>
```

CSS styles.css

```
body {
  background-color: powderblue;
}

h1 {
  color: blue;
}

p {
  color: red;
}
```

CSS sintassi

La sintassi di base dei CSS è data dalla dichiarazione di un **selector**, seguito da una lista di **proprietà** e del loro relativo **valore**

```
selector {
  proprietà: valore;
}
```

In questo caso il selettore è **body** e la proprietà impostata è **background-color** seguita dal suo valore **red**

```
body {
  background-color: red;
}
```

CSS selector

I **selector** permettono di identificare in modo preciso gli elementi HTML che vogliamo personalizzare.

Esistono 5 categorie diverse di **selector**

selector semplici : selezionano gli elementi in base a nome, id, classe
selector combinatori o di relazione : selezionano gli elementi in base alla loro relazione
selector di pseudo-classe : selezionano gli elementi in base a uno stato
selector di pseudo-elementi : selezionano e definiscono lo stile di una parte di un elemento
selector di attributo : selezionano gli elementi in base a un attributo o al valore di un attributo

CSS selector semplici

I **selector** semplici selezionano gli elementi in base a nome, id, classe

```
tag {
  color: green;
}
#idtag {
  color: red;
}
.classenome {
  color: magenta;
}
tag.classenome {
  color: magenta;
}
```

CSS selector universale

Per convenzione esiste il selector * che indica che le proprietà indicate devono essere applicate a qualsiasi **tag**

```
* {
  color: green;
}
```

In questo modo, qualsiasi elemento contenuto in pagina, avrà una colorazione di default impostata sul verde

CSS selector raggruppamenti

Per ridurre la prolissità dei CSS è stata introdotta la sintassi per raggruppamento che permette di mettere, in un'unica dichiarazione, più direttive CSS. Per questo motivo scrivere

```
h1 {
  color: green;
}
h2 {
  color: green;
}
```

Equivale a scrivere

```
h1, h2 {
  color: green;
}
```

CSS selector combinator

I **selector combinatori :** selezionano gli elementi in base alla loro relazione. Per determinare la relazione viene usato un **combinator**. I combinator possono essere di 4 tipi

discendente (spazio)
figlio (>)
fratelli adiacenti (+)
fratelli generali (~)

CSS selector combinator

discendente (spazio)
div p { background-color: red; }

figlio (>)
div > p { color: white; }

fratelli adiacenti (+)
h3 + span { color: white; }

fratelli generali (~)
h2 ~ h3 { border: 1px solid black; }

CSS selector pseudo classe

Una pseudo classe identifica uno stato speciale di un tag.

La sintassi di utilizzo è simile alla sintassi base, con l'aggiunta di : e il tipo di pseudoclasse

```
selector:pseudoclasse {
  proprietà: valore;
}
```

CSS selector pseudo classe

Esistono una trentina di pseudoclassi.

Di seguito alcune classi

:hover è attiva quando il puntatore del mouse è sopra l'elemento
:focus un input che riceve il fuoco
:read-only un input con l'attributo **readonly**

Sul sito dei developer Mozilla è possibile averne un elenco completo

https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

CSS selector pseudo elemento

Un pseudo elemento viene utilizzato per applicare uno stile a una parte specifica di un elemento.

La sintassi di utilizzo è simile alle pseudo classi, con l'aggiunta di un doppio : e il tipo di pseudo elemento

```
selector::pseudoelemento {
  proprietà: valore;
}
```

CSS selector pseudo elemento

Un pseudo elemento viene utilizzato per applicare uno stile a una parte specifica di un elemento.

::after prima dell'elemento
::before dopo l'elemento
::first-child è il primo elemento di una lista

CSS attribute selector

Tramite i selector di attributi è possibile applicare uno stile agli elementi HTML che hanno attributi o valori di attributo specifici.

```
selector[attributo] {
  proprietà: valore;
}

selector[attributo=valore] {
  proprietà: valore;
}
```

CSS attribute selector

Seleziono solo il tag **p** che ha un attributo chiamato **test1**

```
p[test1] {
  color: red;
}
```

Seleziono solo il tag **p** che ha un attributo chiamato **test2** col valore **pippo**

```
p[test2=pippo] {
  color: green;
}
```

CSS attribute selector

Seleziono un attributo che contiene la parola valore

```
selector[attributo~=valore] {
  proprietà: valore;
}
```

Seleziono un attributo che inizia per valore (uguale o uguale seguito dal trattino)

```
selector[attributo|=valore] {
  proprietà: valore;
}
```

CSS attribute selector

Seleziono un attributo che inizia valore

```
selector[attributo^=valore] {
  proprietà: valore;
}
```

Seleziono un attributo che finisce per valore

```
selector[attributo$=valore] {
  proprietà: valore;
}
```

CSS attribute selector

Seleziono un attributo che contiene la sottostringa valore

```
selector[attributo*=valore] {
  proprietà: valore;
}
```

CSS commenti

All’interno di un CSS è possibile inserire dei commenti
I commenti sono multiriga, iniziano con /* e terminano con */

```
selector {
  proprietà: valore; /* commento */
}
```

CSS colori

All’interno di un CSS è possibile referenziare dei colori

I colori possono essere specificati usando il nome predefinito del colore o le sintassi RGB, HEX, HSL, RGBA e HSLA

CSS come usare i colori

Background

Lorem ipsum

Foreground

Lorem ipsum

Border

Lorem ipsum

CSS background

I background possono avere varie proprietà di personalizzazione.

Colore

Lorem ipsum

Opacità/trasparenza. Può assumere un valore compreso tra 0,0 e 1,0. Più basso è il valore, più è trasparente:

Lorem ipsum

CSS background

background-image

Lorem ipsum

background-pepeat

Lorem ipsum

background-position

Lorem ipsum

CSS background

background-attachment

Lorem ipsum

Lorem ipsum

CSS border

Le proprietà **border** consente di specificare lo stile, la larghezza e il colore del bordo di un elemento.

border può essere usata in modo compatto o specificandone le singole caratteristiche

CSS border-style

border-style indica lo stile del bordo

dotted - bordo punteggiato
dashed - bordo tratteggiato
solid - bordo continuo
double - doppio bordo
groove - bordo scanalato 3D
ridge - bordo increspato 3D
inset - bordo del riquadro 3D
outset - bordo iniziale 3D
none - nessun bordo
hidden - bordo nascosto

CSS border-style

Da notare che la proprietà può essere indicata
anche 2, 3 o 4 volte con valori diversi.
Se indicata 2 volte: vengono indicati i bordi superiore/inferiore e destro/sinistro.
Se 3: bordo superiore, bordo destro/sinistro e inferiore. Se 4: bordo superiore, destro, inferiore e sinistro.

```
<p style="border-style: dotted;">dotted</p>
<p style="border-style: dashed;">dashed</p>
<p style="border-style: solid;">solid</p>
<p style="border-style: double;">double</p>
<p style="border-style: groove;">groove</p>
<p style="border-style: ridge;">ridge</p>
<p style="border-style: inset;">inset</p>
<p style="border-style: outset;">outset</p>
<p style="border-style: none;">none</p>
<p style="border-style: hidden;">hidden</p>
<p style="border-style: dotted solid;">mix1</p>
<p style="border-style: dotted solid dashed;">mix2</p>
<p style="border-style: dotted dashed solid double;">mix3</p>
```

CSS border-width

La proprietà **border-width** indica la grandezza dei 4 bordi.

La grandezza può assumere un valore numerico in px, pt, cm, em o usare uno dei valori predefiniti: thin, medium o thick.

```
<p style="border-style: solid; border-width: 5px;">solid - width: 5px</p>
<p style="border-style: solid; border-width: thin;">solid - thin</p>
<p style="border-style: solid; border-width: medium;">solid - medium</p>
<p style="border-style: solid; border-width: thick;">solid - thick</p>
```

CSS border-color

La proprietà **border-color** indica il colore dei 4 bordi.

Il colore un valore espresso tramite nome, in esadecimale, RGB o HSL

solid - red
solid - aeaeae

CSS border lati

Le proprietà precedenti identificano in modo generale tutti i lati di un bordo. È però possibile indicare, singolarmente, i singoli bordi con la sintassi

border-<lato>-<proprietà>

Dove i lati sono indicati come: **top, left, bottom** e **right**

border-**top**-color: red;
border-**bottom**-width: 10px;
border-**bottom**-style: dotted;

CSS border

Le proprietà precedenti possono essere compresse nell'unica proprietà **border.**
Possiamo quindi specificare le proprietà:

border-style (obbligatoria)
border-width
border-color

all'interno della stessa proprietà:

Border

CSS border-radius

Le proprietà **border-radius** permette di indicare che il bordo deve aver gli angoli arrotondati. All'interno di questa proprietà va indicato il valore di arrotondamento

border-radius: 10px;

Questo valore può essere espresso in pixel o in percentuale

CSS margin

Le proprietà **margin** permette di indicare uno spazio attorno ai bordi.
Anche in questo caso è possibile indicare i singoli bordi sui quali applicare i margini

Margin1

Margin2

CSS margin


hotel-room.css
.air-conditioner {
margin-left: -25px;
}
STARECAT.COM

CSS padding

Le proprietà **padding** permette di indicare uno spazio interno ai bordi.
Anche in questo caso è possibile indicare i singoli bordi sui quali applicare il padding

Padding1

Padding2

CSS height width

Le proprietà **height** e **width** permettono di indicare l'altezza e la larghezza di un elemento.

Queste proprietà possono essere limitate, usando le proprietà corrispondenti **max** e **min**

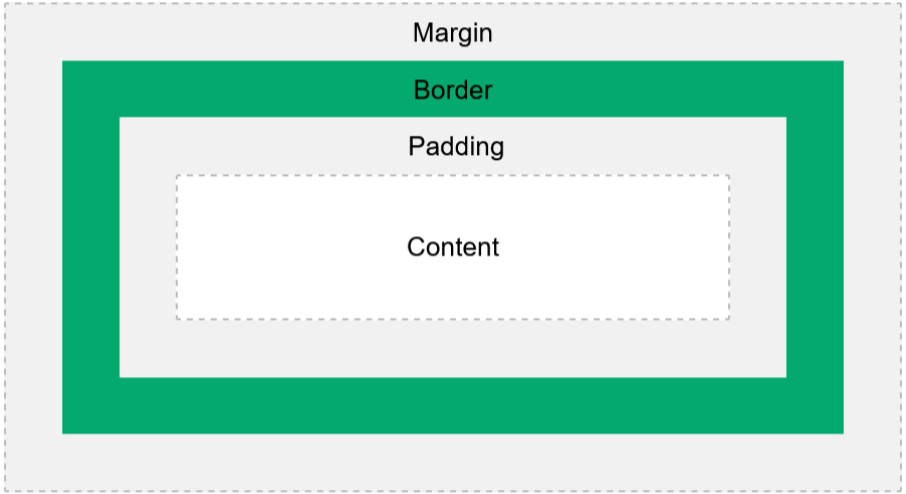
max-width: larghezza massima
min-width: larghezza minima
max-heigth: altezza massima
min-heigth: altezza minima

CSS height width

CSS box model

Il termine "box model" viene utilizzato per indicare il design e layout.

Il box model è il riquadro che avvolge ogni elemento HTML, costituito da **margin**, **border**, **padding** e contenuto.



CSS outline

Esternamente al **border**, senza interferire con il dimensionamento del **margin**, è possibile lavorare con le proprietà do **outline**.

outline-style: ha gli stessi stili di border
outline-color: indica il colore dell'outine
outline-width: la dimensione dell'outline
outline-offset: l'offset rispetto al bordo
outline: la proprietà compressa

CSS text

Tramite CSS è possibile formattare e dare uno stile ai testi.

Le proprietà utilizzabile sono **text-align** e **direction**

text-align: l'allineamento del testo
text-align-last: l'allineamento del testo dell'ultima riga di un paragrafo
direction: la direzione del testo

CSS text decoration

È possibile dare delle caratteristiche al testo.

text-decoration-line: tipo di linea
text-decoration-color: colore
text-decoration-style: stile
text-decoration-thickness: spessore
text-decoration: proprietà unica

CSS text transformation

Tramite **text-transform** è possibile mettere in maiuscolo, minuscolo l'intero testo o la prima lettera del testo:

text-transform: uppercase;
text-transform: lowercase;
text-transform: capitalize;

CSS text spacing

La spaziatura dei testi, la loro altezza, lo spazio fra parole e caratteri sono tutte caratteristiche che possono essere variate tramite le proprietà:

text-indent
letter-spacing
line-height
word-spacing
white-space

CSS text shadow

Con la proprietà **text-shadow** è possibile impostare delle ombre ai testi:

text-shadow: <orizzontale> <verticale> <sfocatura> <colore>

CSS font

Utilizzare il corretto font in base al sito che si vuole costruire ha una enorme importanza.
I font permettono di dare una impronta distintiva del sito e di far percepire immediatamente lo stile utilizzato.
La prima proprietà usata per i font è:

font-family: tipologia di famiglia

CSS font

Le famiglie generiche utilizzabili in CSS sono

serif: hanno un piccolo tratto ai bordi di ogni lettera per creare formalità ed eleganza.
sans-serif: hanno linee pulite e creano un look moderno e minimalista.
monospace: tutte le lettere hanno la stessa larghezza fissa per creare un aspetto meccanico.
cursive: imitano la scrittura manuale.
fantasy: sono caratteri decorativi/giocosi.

CSS font

Esempi di font

Generic Font Family	Examples of Font Names
Serif	Times New Roman Georgia Garamond
Sans-serif	Arial Verdana Helvetica
Monospace	Courier New Lucida Console Monaco
Cursive	Brush Script MT Lucida Handwriting
Fantasy	Copperplate Papyrus

CSS font web safe

Vista la varietà di sistemi operativi e browser, esiste una convenzione per l'utilizzo di font universalmente utilizzabili all'interno di un browser.
Questi font sono:

Arial (sans-serif)
Verdana (sans-serif)
Helvetica (sans-serif)
Tahoma (sans-serif)
Trebuchet MS (sans-serif)
Times New Roman (serif)
Georgia (serif)
Garamond (serif)
Courier New (monospace)
Brush Script MT (cursive)

CSS font fallback

Per garantire una corretta visualizzazione delle pagine, è buona norma utilizzare la sequenza dichiarativa di font in questo modo

font-family: <font>, <websafe>, <famiglia>

CSS font style size

Nei font è possibile variare anche style e size

font-style: italic; /* italic, normal o oblique */
font-weight: bold; /* bold o normal */
font-size: 40px;

CSS font google

Google mette a disposizione una serie di font direttamente utilizzabili

```
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">

<h1 style="font-family: 'Sofia', sans-serif;">Testo in font Sofia</h1>
```

CSS icone google

Oltre ai font Google mette a disposizione una serie icone liberamente importabili nei nostri progetti

```
<link rel="stylesheet" href="https://fonts.sandbox.google.com/css2?
family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />
<style>
    .material-symbols-outlined {
      font-variation-settings:
        'FILL'0,
        'wght'400,
        'GRAD'0,
        'opsz'48
    }
</style>

<span class="material-symbols-outlined">
    headset_mic
</span>
```

CSS icone awesome

L’utilizzo di icone permette di rendere più intuitive le pagine.
Oltrea alle icone google ci sono molte alternative in rete, come quelle fornite da fontawesome https://fontawesome.com/

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.1/css/all.min.css">

<i class="fa-regular fa-user"></i>
```

Fonte: https://fontawesome.com/icons/

CSS link

Il tag **a** può essere personalizzato in base allo stato del link, utilizzando le proprietà che vengono usati per i testi.
Gli stati condizionabili dei link sono

a:link – link non visitato
a:visited – link visitato
a:hover – quando il mouse si posiziona sopra al link
a:active – un link quando viene cliccato

CSS liste

Le liste html possono essere **ordered** (ol) e **unordered** (ul).
Questi tipi di lista possono essere personalizzate tramite **list-style-type**

list-style-type: circle;
list-style-type: square;
list-style-type: upper-roman;
list-style-type: lower-alpha;

list-style-image: referenzia una immagine da unire alla voce

list-style-image: url('dot.gif');

CSS liste

list-style-position: definisce la posizione del marker di lista, se deve essere interno o esterno alla lista

list-style-position: inside;
list-style-position: outside;

CSS table

Le tabelle hanno un gran numero di personalizzazioni basate sulla loro struttura e caratteristiche delle celle e delle colonne. Le principali personalizzazioni sono fattibili su

border: <caratteristiche di bordo>
padding: 10px; /* su TH e TD */
width: <larghezza: ex 100%>
color: <colore>
height: <larghezza: ex 100px su TH>
border-collapse: collapse; /* su table condensa il bordo in uno solo */
text-align: center; /* su TD con center, left, right */
tr:hover{ backgroud-color: red; } /* per evidenziare la riga corrente */
tr:nth-child(even) {background-color: grey;} /* Alternare la righe */

CSS display

La proprietà **display** è la più importante per il controllo del layout

Ogni elemento HTML ha un valore di visualizzazione predefinito

Per la maggior parte degli elementi il default è **block** o **inline**

display a **none** permette di nascondere degli elementi a video

CSS display

Esempi di **display** block sono

div, h1, h2, …, h6, p, form, header, footer e section

Esempi di **display** a **inline** sono

span, a, img

CSS max-width

width e **max-width** ci permettono di definire la dimensione di un elemento

Mentre **width** è una imposizione, **max-width** è una indicazione della massima larghezza utilizzabile per un particolare elemento

Unita alle proprietà **margin:auto** possiamo facilmente gestire una centratura dell'elemento all'interno della larghezza di pagina

CSS position

La proprietà **position** indica il tipo di posizionamento utilizzato per un elemento

- static
- relative
- fixed
- absolute
- sticky

Ogni posizionamento ha delle proprie caratteristiche

CSS position static

L’impostazione **static** è il valore predefinito degli elementi HTML

Gli elementi **static** non sono interessati dalle proprietà top, bottom, left e right

Un elemento con posizione **static** non è posizionato secondo il normale flusso della pagina

CSS position relative

Un elemento **relative** è posizionato rispetto alla sua posizione normale

L'impostazione delle proprietà **top, bottom, left** e **right** sposterà l'elemento dalla sua posizione normale

Gli altri contenuti non verranno adattati per adattarsi a eventuali spazi vuoti lasciati dall'elemento

CSS position fixed

Un elemento **fixed** è posizionato rispetto al viewport e rimane sempre nella stessa posizione anche se la pagina viene fatta scorrere

Le proprietà **top, bottom, left** e **right** servono a posizionare l'elemento

Un elemento fisso non lascia lo spazio vuoto nella pagina in cui sarebbe stato normalmente posizionato

CSS position absolute

Un elemento **absolute** è posizionato rispetto all'antenato posizionato più vicino

Se un elemento posizionato **absolute** non ha antenati posizionati, utilizza il corpo del documento e si sposta insieme allo scorrimento della pagina

Gli elementi posizionati assoluti vengono rimossi dal flusso normale e possono sovrapporsi ad altri elementi

CSS position sticky (appiccicoso)

Un elemento **sticky** è posizionato in base alla posizione di scorrimento dell'utente

Un elemento **sticky** si alterna tra **relative** e **fixed**, a seconda della posizione di scorrimento

Viene posizionato in modo **relative** fino a quando una determinata posizione di offset non viene raggiunta nella finestra, quindi si attacca in posizione come **fixed**

CSS z-index

Gli elementi all'interno di una pagina possono sovrapporsi a causa del loro posizionamento.

Per poter definire la priorità di ogni elemento, è possibile assegnare la proprietà **z-index** seguita da un numero, più è basso il numero, più l'elemento verrà messo di sfondo. Più è alto più sarà messo in primo piano.

CSS overflow

La proprietà **overflow** controlla cosa succede al contenuto che è troppo grande per adattarsi a un'area.
I valori che può assumere sono:

visible – Rappresenta il default: il contenuto non è tagliato ed esce dall'area
hidden – Viene ritagliato e visualizzato solo quanto presente in area
scroll – Viene ritagliato ed aggiunta una barra per scorrere il contenuto
auto – Come scroll, ma le barre sono aggiunte solo se necessarie

CSS float

La proprietà **float** definisce come un elemento deve fluttuare.
I valori possibili sono:

left – L'elemento fluttua a sinistra del suo contenitore
right – L'elemento fluttua a destra del suo contenitore
none – Default: l'elemento non è mobile, verrà visualizzato dove si trova.
inherit – L'elemento eredita il valore float del suo genitore

CSS clear

La proprietà **clear** obbliga il prossimo elemento a posizionarsi sotto all'elemento corrente

none – rappresenta il default e non sposta l'elemento sotto agli elementi **float**
left – L'elemento è messo sotto agli elementi **float left**
right – L'elemento è messo sotto agli elementi **float right**
both – L'elemento è messo sotto agli elementi **float right** o **left**
inherit – L'elemento eredita il valore **clear** del suo genitore

CSS allineamenti

Esistono varie tecniche per poter allineare un elemento.

Per quanto riguarda l'allineamento orizzontale è possibile usare la proprietà **margin** col valore ad **auto**

Per i testi è possibile utilizzare la proprietà **text-align** col valore **center**

Per le immagini occorre indicare le proprietà **display** a **block** e **margin-left** e **margin-right** ad **auto**

Il **padding** può invece essere usato per una centratura verticale

CSS !important

Tramite la regola **!important** è possibile sovrascrivere qualsiasi regola precedente, dando priorità alla corrente.

Questo è utile se vogliamo dare una importanza ad una certa regola rispetto ad altre

CSS screen resolution

Con l'evoluzione degli apparati che si collegano ad internet è difficile definire un vero e proprio standard di risoluzione video

Per questo motivo ogni sviluppatore di librerie CSS ha adottato nel corso del tempo degli standard che col tempo si sono evoluti

Il comportamento minimo che si è sempre cercato di seguire è però stato quello di indirizzare correttamente almeno 4 tipi di visualizzazioni

- Schermi piccoli: cellulari
- Schermi medi: tablet o cellulari a display orizzontale
- Schermi grandi: PC
- Schermi molto grandi: schermi full HD

CSS screen resolution

Un buon compromesso può essere l’uso delle seguenti media query

```
// Schermi medio-piccoli (almeno 576px)
@media (min-width: 576px) { ... }

// Schermi medi (almeno 768px)
@media (min-width: 768px) { ... }

// Schermi grandi (almeno 992px)
@media (min-width: 992px) { ... }

// Schermi molto grandi (almeno 1200px)
@media (min-width: 1200px) { ... }
```

https://italia.github.io/bootstrap-italia/docs/organizzare-gli-spazi/introduzione/