Corso CSS3

Impariamo a usare CSS3
conoscendone le basi

Scopo del corso

Queste slide nascono dalla mia curiosità verso CSS3 e dall'esperienza che ho accumulato nel corso degli anni.

La struttura base del corso ripercorre i passi seguiti da w3schools e li arricchisce con nuovi esempi ed una serie di slide in italiano.

CSS - di cosa si tratta?

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 – editor

- Codepen.io
- Notepad
- Notepad++
- VisualStudio Code
- Codespace di GitHub

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

Dal gruppo Coding Chess di Facebook

CSS – esempio

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

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

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

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 inserimento in pagina

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

```
<head>
    <style>
        body {
            background-color: powderblue;
        }
        h1 {
            color: blue;
        }
        p {
            color: red;
        }
    </style>
</head>
```

CSS Esterno

```html
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="styles.css">
    </head>
    <body>
        <h1>Intestazione</h1>
        <p>Paragrafo</p>
    </body>
</html>
```

CSS styles.css

```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;
}

#id {
    color: red;
}

.classe {
    color: magenta;
}

tag.classe {
    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. Ad esempio:

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

Equivale a scrivere

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

CSS selector combinator

I **selector combinator** 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 - tipologie

discendente (spazio) - un qualsiasi discendente

```
div p { background-color: red; }
```

figlio (>) - solo i figli diretti

```
div > p { color: white; }
```

CSS selector combinator - fratelli

fratelli adiacenti (+) - solo il primo fratello

```
h3 + span { color: white; }
```

fratelli generali (~) - tutti i fratelli

```
h2 ~ h3 { border: 1px solid black; }
```

CSS selector pseudo classe

Una pseudo classe identifica uno stato speciale di un tag: p.e. hover

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: esempi

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: p.e. ::after

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 - esempi

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

::after dopo l'elemento
::before prima dell'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 - esempi

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 **foobar**

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

CSS attribute selector - 1

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 - 2

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 - 3

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

```
<div style="background-color:red;">Lorem ipsum</div>
```

Foreground

```
<div style="color:blue;">Lorem ipsum</div>
```

Border

```
<div style="border:2px solid red;">Lorem ipsum</div>
```

CSS background - 1

I background possono avere varie proprietà di personalizzazione.

Colore

```
<div style="background-color:red;">Lorem ipsum</div>
```

Opacità/trasparenza. Può assumere un valore compreso tra 0.0 e 1.0. Più basso è il valore, più è trasparente:

```
<div style="background-color:red; opacity: 0.3;">Lorem ipsum</div>
```

CSS background - 2

background-image

```
<div style="background-image: url(gattino.gif);">Lorem ipsum</div>
```

background-repeat

```
<div style="background-image: url(gattino.gif);
background-repeat: repeat-x;">Lorem ipsum</div>
```

background-position

```
<div style="background-image: url(gattino.gif);
background-repeat: no-repeat; background-position: right top;">Lorem ipsum</div>
```

CSS background

background-attachment

fixed - il background è attaccato al viewport

scroll - il background è attaccato al contenuto

```
<div style="background-image: url(gattino.gif);
background-attachment:fixed;">Lorem ipsum</div>

<div style="background-image: url(gattino.gif);
background-attachment:scroll;">Lorem ipsum</div>
```

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 - note

Da notare che la proprietà può essere indicata anche 2, 3 o 4 volte con valori diversi.

Se 2: 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.

Border - 1

```
<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>
```

Border - 2

```
<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

```
<p style="border-style: solid; border-color: red;">solid - red</p>

<p style="border-style: solid; border-color: #aeaeae;">solid - #aeaeae</p>
```

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> : bordo - lato e 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 - proprietà unica

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:

```
<p style="border: 3px solid red;">Border</p>
```

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

```
<p style="margin: 16px 10px 0 10px;">Margin1</p>
<p style="margin-top:16px; margin-left:10px; margin-right:10px;">Margin2</p>
```

CSS margin - esempio

https://me.me/i/margin-left-25px-everyone-should-know-css-a2abdd33f2b5499ba28fa8afab9dfc2b

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

```
<p style="padding: 0px 10px;">Padding1</p>

<p style="padding-top:0px; padding-left:10px; padding-right:10px;">Padding2</p>
```

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**

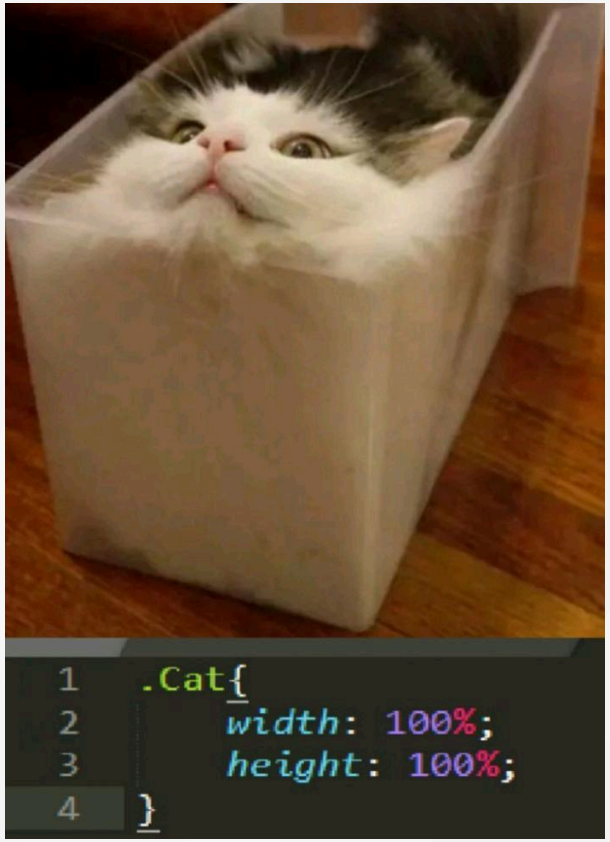
max-width: larghezza massima

min-width: larghezza minima

max-height: altezza massima

min-height: altezza minima

CSS height width - esempio



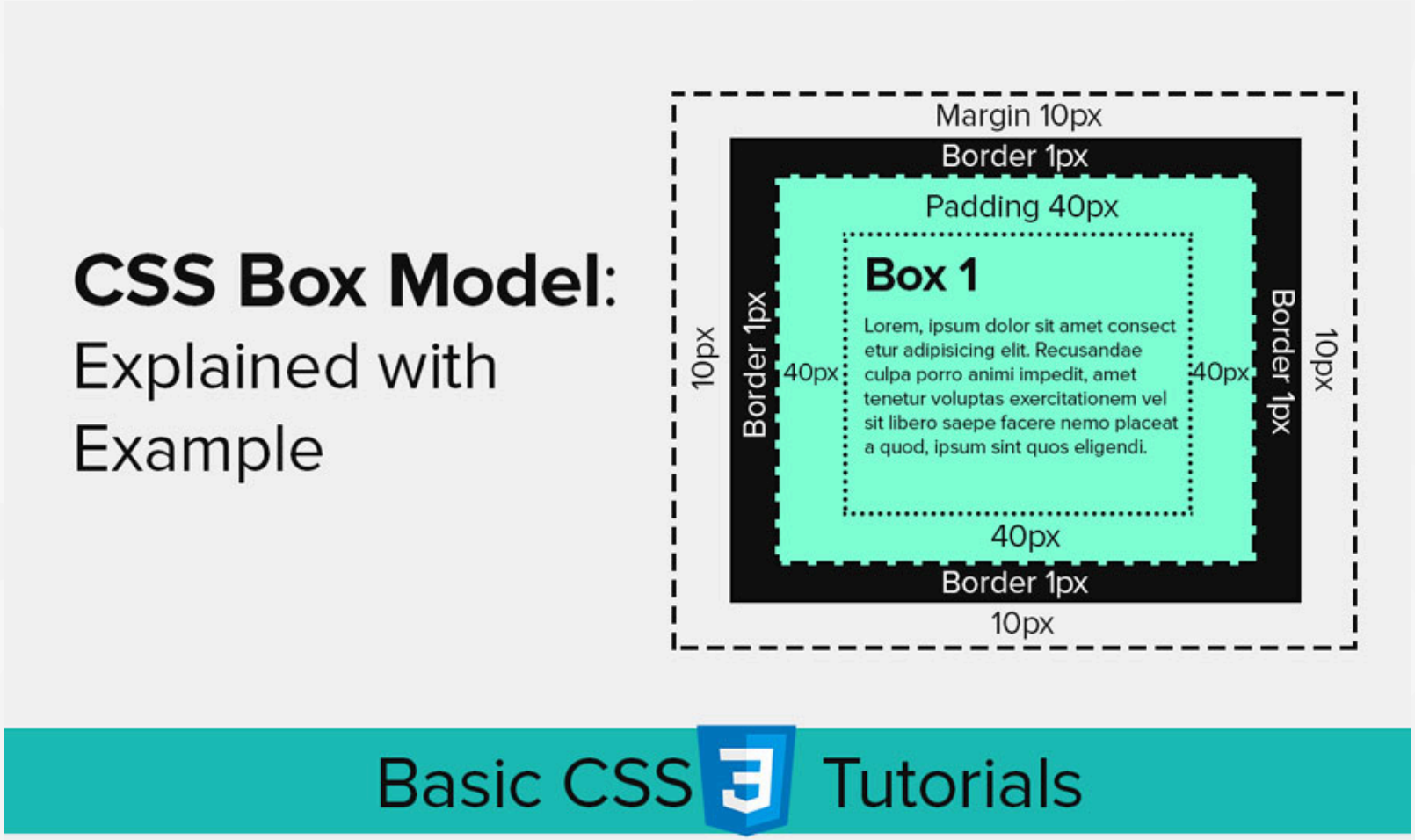
https://programmerhumour.tumblr.com/post/643187358740725760/css-cat

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.



https://www.webcodzing.com/css-box-model/

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'outline

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 : l'indentazione del testo

letter-spacing : lo spazio tra le lettere

line-height : l'altezza della riga

word-spacing : lo spazio tra le parole

white-space : la gestione degli spazi bianchi

CSS text shadow

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

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

Alcuni esempi di ombre sono disponibili a questo indirizzo

https://designshack.net/articles/css/12-fun-css-text-shadows-you-can-copy-and-paste/

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 - famiglie

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

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, Verdana, Helvetica, Tahoma, Trebuchet MS (sans-serif)
Times New Roman, Georgia, 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.

Oltre a 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 usate per i testi.

Gli stati 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 - 2

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{ background-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 - 2

TAG con **display** a **block**

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

TAG con **display** a **inline**

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 Gradienti

La proprietà **background-image** può assumere delle tonalità gradienti che consentono di visualizzare transizioni uniformi tra due o più colori specificati.

I CSS definiscono tre tipi di gradienti:

- Gradienti lineari (vanno da giù/su/sinistra/destra/diagonalmente)
- Gradienti radiali (definiti dal loro centro)
- Gradienti conici (ruotati attorno a un punto centrale)

```
background-image: linear-gradient(red, black);
```

CSS Transition

Le transizioni CSS ti consentono di modificare i valori delle proprietà lentamente, entro una certa durata

```
transition
transition-delay
transition-duration
transition-property
transition-timing-function
```

CSS utilizzo delle Transition

Per creare un effetto di **transition**, occorre indicare due cose

- su quale proprietà CSS aggiungere l'effetto
- la sua durata

CSS Transition - esempi1

Transition della sola larghezza

```
#div {
    border: 1px solid black;
    width: 100px;
    height: 100px;
    transition: width 2s;
}
#div:hover {
    width: 200px;
    height: 200px;
    border: 2px dotted red;
}
```

CSS Transition - esempi2

Transition di larghezza e altezza

```
#div {
    border: 1px solid black;
    width: 100px;
    height: 100px;
    transition: width 2s, height 2s;
}
#div:hover {
    width: 200px;
    height: 200px;
    border: 2px dotted red;
}
```

CSS Transition - esempi3

Transition di tutte le proprietà

```
#div {
    border: 1px solid black;
    width: 100px;
    height: 100px;
    transition: all 2s;
}
#div:hover {
    width: 200px;
    height: 200px;
    border: 2px dotted red;
}
```

CSS Animation

Animation permette la creazione di animazioni senza l'uso di Javascript

Un'animazione consente a un elemento di cambiare gradualmente da uno stile all'altro.

Si possono modificare tutte le proprietà CSS, tutte le volte che si vuole.

Per utilizzare animation occorre specificare alcuni **@keyframe** per l'animazione.

I **@keyframe** contengono gli stili che l'elemento dovrà avere.

CSS Animation - proprietà

- **@keyframes** : lo stile dell'elemento
- animation-name : nome del keyframe
- animation-duration : durata
- animation-delay : ritardo
- animation-iteration-count : il numero di esecuzione di keyframe o infinite
- animation-direction : la direzione dell'animazione
- animation-timing-function : la velocità dell'animazione
- animation-fill-mode : lo stile per l'elemento di destinazione quando l'animazione non viene riprodotta
- animation : la proprietà cumulativa

CSS Animation - esempi - rotazione

Viene impostato il keyframe **rotation** da eseguire in 10 secondi, all'infinito, in modo lineare.

```
.rotazione{
    animation: rotation 10s infinite linear;
}

@keyframes rotation {
    from {
        transform: rotate(0deg);
    }
    to {
        transform: rotate(359deg);
    }
}
```

Esempi: https://freefrontend.com/css-animation-examples/

CSS Image Filter

Sulle immagini è possibile inserire un **filter**, in modo che l'immagine venga visualizzata con uno stile diverso.

I filtri possibili sono:

```
filter: blur(5px);
filter: brightness(0.4);
filter: contrast(200%);
filter: drop-shadow(16px 16px 20px blue);
filter: grayscale(50%);
filter: hue-rotate(90deg);
filter: invert(75%);
filter: opacity(25%);
filter: saturate(30%);
filter: sepia(60%);
```

https://caniuse.com/css-filters

CSS Resize

La proprietà **resize** permette di dare all'utente la possibilità di ridimensionare un elemento

In questo modo si aumenta l'interazione che gli utenti possono avere con gli elementi della pagina.

https://caniuse.com/css-resize

CSS var

I CSS possono dichiarare delle variabili, utilizzabili successivamente all'interno delle proprietà

La dichiarazione di una variabile richiede l'utilizzo di una etichetta preceduta da un doppio meno e la dichiarazione all'interno del selector :root

```
:root {
  --rosso: #ff0000;
  --bianco: #ffffff;
}

body {
    background-color: var(--rosso);
    color: var(--bianco);
}
```

https://caniuse.com/css-variables

CSS 2D transform

Le transform CSS ti consentono di spostare, ruotare, ridimensionare e inclinare gli elementi.

Tutte queste trasformazioni avvengono tramite la proprietà

```
transform
```

https://caniuse.com/transforms2d

CSS 2D transform - esempi

Il metodo translate() muove un elemento dalla sua posizione rispetto all'asse X e Y

```
transform: translate(80px, 10px);
```

Il metodo rotate() ruota un elemento in senso orario o antiorario in base a un grado

```
transform: rotate(10deg);
transform: rotate(-10deg);
```

Il metodo scale() aumenta o diminuisce la dimensione di un elemento per larghezza e altezza, mentre i metodi scaleX() e scaleY() lavorano su una sola dimensione

```
transform: scale(3, 2);
transform: scale(0.8, 0.2);
transform: scaleX(3);
transform: scaleY(4);
```

CSS 2D transform - esempi2

Il metodo skew() inclina un elemento lungo gli assi X e Y, mentre i metodi skewX() e skewY() lavorano su un singolo asse

```
transform: skew(10deg, 10deg);
transform: skewX(10deg);
transform: skewY(10deg);
```

Il metodo matrix() combina tutte le trasformazioni 2D in una sola.

I parametri seguono il seguente schema

```
matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())
```

```
transform: matrix(1, -0.3, 0, 1, 0, 0);
```

CSS 3D transform

CSS supporta anche le trasformazioni 3D, con metodi che permettono le rotazioni rispetto all'asse X, Y o Z

```
transform: rotateX(100deg);
transform: rotateY(150deg);
transform: rotateZ(110deg);
```

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 - esempio

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/

https://caniuse.com/css-mediaqueries

CSS nesting

le ultime evoluzioni e gli aggiornamenti del linguaggio CSS hanno introdotto la possibilità di nidificare le regole CSS.

Cosa significa "nidificare" le regole CSS?

In passato, quando si scriveva codice CSS, era necessario ripetere i selettori per ogni regola che si voleva applicare a un elemento specifico o ai suoi discendenti. Questo portava spesso a codice ridondante e difficile da leggere e mantenere.

Con il nesting, invece, è possibile scrivere le regole CSS in modo più gerarchico e organizzato, raggruppando le regole relative a un elemento all'interno del selettore principale. Questo rende il codice più compatto, leggibile e intuitivo.

CSS prima e dopo

```css
.contenitore {
  border: 5px solid black;
}

.contenitore p {
  color: red;
  font-weight: bold;
}
```

```css
.contenitore {
  border: 5px solid black;

  p {
    color: red;
    font-weight: bold;
  }
}
```

https://caniuse.com/css-nesting

CSS @scope

La regola @**scope** in CSS è uno strumento potente per:

- Creare confini di stile (style boundaries).
- Prevenire conflitti di stile.
- Costruire CSS modulare e mantenibile.
- Indirizza delle parti specifiche del DOM.

È una funzionalità che vale la pena imparare man mano che diventa più ampiamente supportata.

https://caniuse.com/css-cascade-scope

CSS @scope sintassi

Sintassi

```
@scope (<radice-ambito>) [to (<limite-ambito>]) {
  /* Regole CSS che si applicano all'interno dell'ambito */
}
```

CSS @scope - esempio

```
Non funziona con FireFox
<div class="card">
    <p>Questo è un paragrafo all'interno della card.</p>
    <div class="nested">
        <p>Questo è un paragrafo annidato.</p>
    </div>
    <div class="limit">
        <p>Questo è un paragrafo fuori dal limite.</p>
    </div>
</div>
<p>Questo è un paragrafo al di fuori della card.</p>
```

```
@scope (.card) to (.limit) {
  p {
    color: red;
    font-weight: bold;
  }
}
```

CSS @layer

La regola @layer nel CSS è uno strumento potente per poter:

- Gestire i conflitti di stile.
- Organizzare il codice CSS in modo modulare.
- Controllare la specificità delle regole.
- Controllare l'ordine del CSS
- Al momento la feature non è molto diffusa ma verrà presto supportata

https://caniuse.com/css-cascade-layers

CSS @layer esempio

Non rispetto la sequenza delle definizioni e cambio le priorità

```
@layer components;
@layer base;

@layer base {
    body {
        background-color: lightgray;
    }
}

@layer components {
    body {
        background-color: lightgreen;
    }
}
```

CSS Color scheme

La media query @media (prefers-color-scheme: ...) è fondamentale per creare siti che rispettino lo schema colore degli utenti.
Ci permette di migliorare la user experience, aumentare l'accessibilità e creare siti web più moderni e personalizzati.
Usando le variabili CSS è possibile scegliere lo schema colore e alternarlo.

https://caniuse.com/?search=prefers-color-scheme

CSS Color scheme - esempio

```
:root {
  --background-color: white;
  --text-color: black;
}

@media (prefers-color-scheme: dark) {
  :root {
    --background-color: #333;
    --text-color: white;
  }
}

body {
  background-color: var(--background-color);
  color: var(--text-color);
}
```

CSS :is e :has

Gli pseudo-selettori **:is** e **:has** sono due potenti strumenti introdotti nelle versioni più recenti di CSS per semplificare e migliorare la selezione degli elementi.

:is è un selector di raggruppamento

:has è un selector di matching fra discendenze

CSS :is

:is

Il selettore :is (precedentemente noto come :matches) permette di raggruppare più selettori in uno solo, riducendo la ripetizione del codice e migliorando la leggibilità. È utile quando si desidera applicare lo stesso stile a diversi selettori

```
:is(selector1, selector2, selector3) {
  proprietà: valore;
}
```

```
:is(h1, h2, h3) {
  color: red;
}
```

https://caniuse.com/css-matches-pseudo

CSS :has

:has

Il selettore :has permette di selezionare un elemento se contiene un determinato discendente che corrisponde al selettore specificato. È particolarmente utile per applicare stili a un elemento in base ai suoi contenuti.

```
selector:has(selector) {
  proprietà: valore;
}
```

```
div:has(p) {
  border: 1px solid black;
}
```

https://caniuse.com/css-has

Select

Da Chrome 135 è possibile personalizzare la select

```
appearance: base-select;
```

Precedentemente non era possibile mettere codice HTML nelle select, cambiando l'appearance ora è possibile.

https://developer.chrome.com/blog/a-customizable-select?hl=it

https://caniuse.com/?search=base-select

text-wrap: balance

Tenta di equilibrare il testo distribuendolo in modo armonico fra le righe, evitando di avere righe con contenuti più pieni e righe con contenuti più rarefatti.

https://caniuse.com/?search=text-wrap%3A balance

Content-visibility

Questa proprietà avvisa il browser di non procedere al rendering degli elementi fuori dalla viewport.

Questo approccio permette un caricamento più veloce della pagina e una migliore esperienza utente, in quanto il browser non deve calcolare lo stile e il layout degli elementi invisibili.

https://drafts.csswg.org/css-contain-2/#content-visibility
https://web.dev/articles/content-visibility?hl=it
https://caniuse.com/css-content-visibility

Fonti

https://www.w3schools.com : argomenti ed idee per esempi

https://it.wikipedia.org : definizioni e argomenti

https://www.w3.org/Style/CSS/specs.en.html : specifiche CSS3

https://github.com/bradtraversy/50projects50days : 50 Projects in 50 Days - HTML/CSS and JavaScript

Ogni immagine inserita riporta la fonte

Disclaimer

Questo materiale è stato realizzato con le seguenti modalità:

- Contenuto testuale
  Redatto attraverso sistemi di AI per la generazione della bozza iniziale, successivamente rielaborato, verificato e integrato manualmente dall'autore.
- Elementi grafici
  L'immagine di sfondo è stata generata tramite Haikei.app. Eventuali altri elementi visivi derivano da banche immagini royalty-free o creazioni originali.
- Ricerche
  I dati e le informazioni citate sono state raccolte da fonti pubbliche accessibili online, selezionate e contestualizzate in modo critico dall'autore.

La direzione intellettuale, le scelte contenutistiche e l'accuratezza delle informazioni restano sotto la piena responsabilità dell'autore.