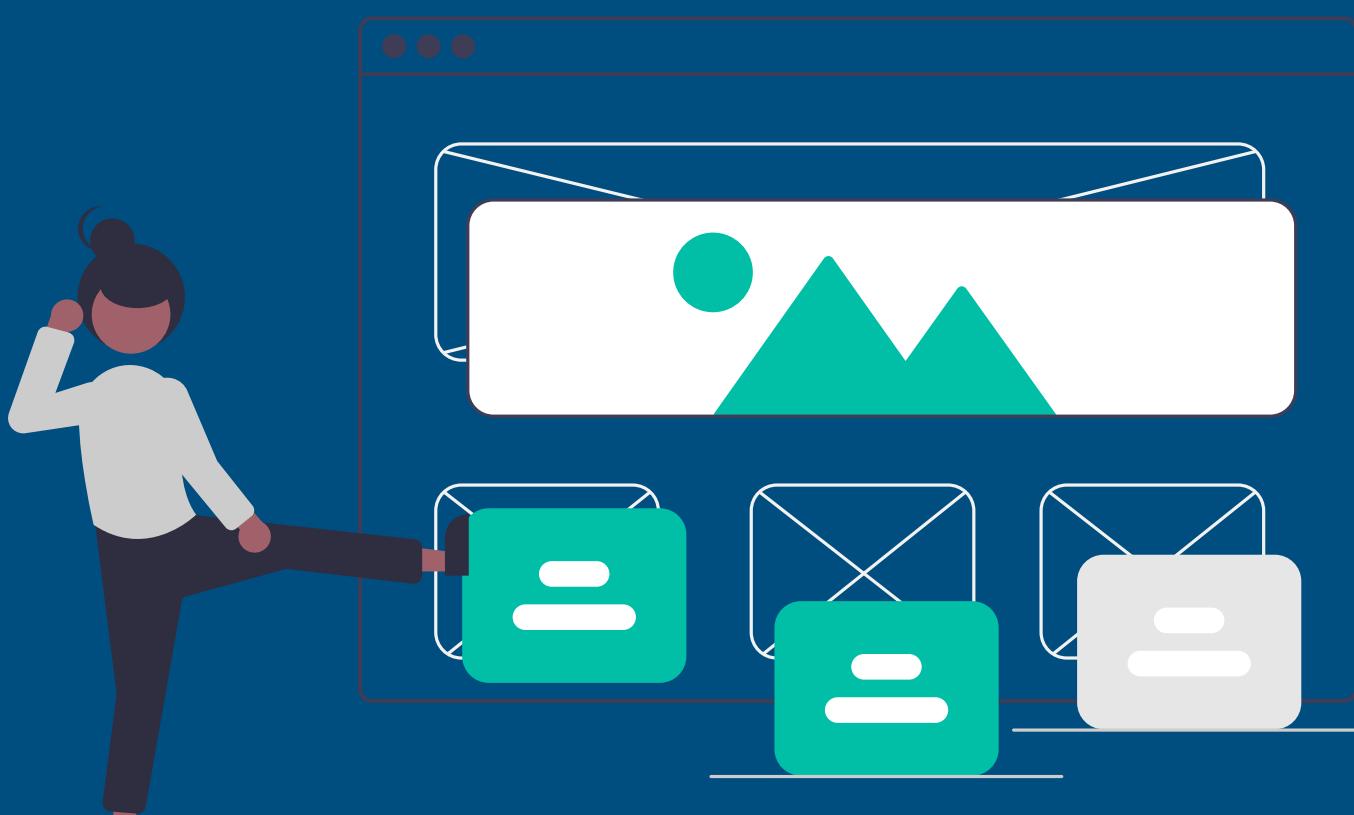




CSS

Web fundamentals



Cos'è il CSS	3
La sintassi	4
Utilizzo del CSS	6
Selettori CSS	10
Specificità	17
I commenti	20
Unità di misura	21
Colori	23
Sfondi	26
Font e testi	30
Link	34
Box Model	37
Posizioni	44
Display	52
Visibility e Opacity	54
Liste	56
Pseudo Classi	59
Pseudo Elementi	64
!important	71
Transform	72
Object-fit	76
Transition	78
Animation	81
Media query	84
Flexbox	87
Organizzare i file CSS	95
Alcuni consigli	96



Cos'è il CSS

Il CSS è un linguaggio che ti permette di definire l'aspetto grafico delle pagine web, come colori, font, dimensioni, posizioni e le animazioni degli elementi.

Il CSS si usa in combinazione con l'HTML, che è il linguaggio che crea la struttura e il contenuto delle pagine web. Il CSS è utile perché ti permette di separare la forma dal contenuto, rendendo le pagine web più leggere, più facili da mantenere e più adattabili a diversi dispositivi e risoluzioni. Il CSS è anche un linguaggio creativo, che ti permette di esprimere il tuo stile e la tua personalità nel web design.

Nei prossimi moduli imparerai come sviluppare in CSS, le nozioni base ed alcuni concetti avanzati che ti permetteranno di definire colori, sfondi, testi, dimensioni, animazioni e tanto altro ancora.



La sintassi

Immagina il CSS come un insieme di istruzioni che indicano al browser come presentare l'aspetto della tua pagina web, come colori, sfondi, caratteri, spazi, ecc ...

L'insieme di istruzioni è una **regola di stile** che è composta da un selettore e da un blocco di dichiarazioni CSS all'interno delle parentesi graffe.

```
selettore {  
    proprietà: valore;  
}
```

Il **selettore** identifica gli elementi HTML ai quali si desidera associare lo stile.

La **proprietà** è un attributo che specifica un aspetto o un comportamento di un elemento HTML. Ogni proprietà è associata ad un **valore** che ne determina l'applicazione.

Ad esempio, per cambiare colore del testo dei paragrafi useremo l'istruzione seguente:

```
● ● ●  
p {  
    color: blue;  
}
```

Il selettore p seleziona tutti gli elementi <p> della pagina web e specifica al browser di rendere il testo di tutti i paragrafi di colore blu.



Le proprietà CSS sono molteplici e controllano vari aspetti del design e della presentazione di un documento HTML, alcuni esempi sono `font-size` per impostare la dimensione del carattere, `background-color` per definire il colore di sfondo di un elemento, `text-align` per l'allineamento del testo all'interno di un elemento e tanti altri ancora.

Questi sono solo alcuni esempi di proprietà CSS. Le proprietà possono essere combinate in regole di stile per personalizzare l'aspetto e il comportamento degli elementi HTML su una pagina.



Utilizzo del CSS

Per definire le regole CSS all'interno di una pagina HTML puoi utilizzare modi diversi: **css incorporato** in una pagina HTML, **css inline** specificando il css per singolo elemento HTML o attraverso un **foglio di stile esterno**.

Vediamo nel dettaglio alcuni metodi:

CSS incorporato in pagina

Il CSS incorporato è definito nella sezione `<style>` all'interno del tag `<head>` del documento HTML. Può essere utilizzato per applicare stile a più elementi sulla stessa pagina HTML.



```
<!DOCTYPE html>
<html>
<head>
    <title>Embedded CSS Example</title>
    <style>
        h1 {
            color: red;
            text-align: center;
        }

        p {
            font-size: 18px;
            color: purple;
        }
    </style>
</head>
<body>
    <h1>Questo è un titolo con stile incorporato</h1>
    <p>Questo è un paragrafo con stile incorporato.</p>
</body>
</html>
```

In questa pagina HTML abbiamo definito il colore del titolo `h1` in rosso e allineato il suo contenuto testo al centro. Inoltre abbiamo definito la dimensione del font del paragrafo a `18px` ed il suo colore `purple`.



Css in linea

Il CSS in linea (inline CSS), viene definito direttamente nell'elemento HTML utilizzando l'attributo `style`.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Inline CSS Example</title>
    </head>
    <body>
        <h1 style="color: red; text-align: center">
            Questo è un titolo con stile inline
        </h1>
        <p style="font-size: 18px; color: purple">
            Questo è un paragrafo con stile inline
        </p>
    </body>
</html>
```

Anche in questa pagina HTML abbiamo definito lo stesso stile dell'esempio precedente ma attraverso l'inline CSS, definito all'interno del tag HTML.

Foglio di stile esterno

Il foglio di stile esterno è un file separato con estensione `.css` che contiene le regole di stile. Viene collegato al documento HTML utilizzando l'elemento `<link>` nel tag `<head>`.

File `index.html`

```
<!DOCTYPE html>
<html>
    <head>
        <title>External CSS Example</title>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
        <h1>Questo è un titolo con stile esterno</h1>
        <p>Questo è un paragrafo con stile esterno</p>
    </body>
</html>
```



File style.css

```
h1 {  
    color: red;  
    text-align: center;  
}  
  
p {  
    font-size: 18px;  
    color: purple;  
}
```

Utilizzando un foglio di stile esterno, è possibile utilizzare gli stili su più pagine web.

In generale, l'uso di fogli di stile esterni è considerato una pratica consigliata per migliorare la separazione e facilitare la manutenzione del codice.



Esercizio

All'interno di una cartella vuota crea due file `index.html` e `style.css`.

All'interno del file `index.html` definisci una pagina HTML generica e nel `body` inserisci un titolo `h1`, un titolo `h2`, un paragrafo con un testo generico che contiene un link ad una pagina esterna.

Definisci adesso nel file `style.css` i seguenti stili.

Titolo h1 di dimensione 48px, di colore #333333 e il testo in maiuscolo.

Titolo h2 di dimensione 36px, di colore nero e le prime lettere di ogni parola in maiuscolo.

Paragrafo di dimensione 18px, di colore nero. Cambia colore al link con uno a tua scelta, rimuovi lo stile di default del testo e rendilo sottolineato.



Selettori CSS

I selettori CSS permettono di **selezionare elementi** di un determinato tipo e applicare loro uno stile specifico.

In CSS troviamo diversi tipi di selettori, utili in diverse situazioni.

Selettori di tipo

Il selettore di tipo è il più comune e seleziona tutti gli elementi di un tipo specifico, ad esempio, come visto in precedenza, per selezionare tutti i paragrafi `p`.

Possiamo utilizzare i selettori di tipo per selezionare gli elementi HTML come ad esempio `h1`, `h2`, `a`, `div`, ecc...

```
● ● ●  
h1 {  
    font-size: 36px;  
}
```

Questo imposta la dimensione del testo per **tutti gli elementi** `h1` a 36 pixel.

O per applicare stili specifici alle liste `` o `` in questo modo:

```
● ● ●  
ul {  
    font-size: 22px;  
    list-style-type: square;  
}  
  
ol {  
    font-size: 28px;  
    list-style-type: decimal;  
}
```



Questo esempio imposta il tipo di marcatore per le liste non ordinate (`ul`) a quadrati e specifica la grandezza del testo a 22 pixel. Per le liste ordinate invece specifica il marcatore come numero decimale e la grandezza del testo a 28 pixel.

Un altro esempio è il selettori di tipo per stili specifici sui link `<a>`, ad esempio:

```
● ● ●  
a {  
    text-decoration: none;  
    color: #3366cc;  
}
```

Questa istruzione rimuove la sottolineatura e imposta il colore del testo per tutti i link.

In generale attraverso i selettori di tipo possiamo selezionare gli elementi HTML che preferiamo, questi sono solo alcuni esempi di come i selettori di tipo possono essere utilizzati per selezionare e stilizzare specifici tipi di elementi HTML nelle pagine web.

Selettori Universali

Il selettori universale corrisponde a tutti gli elementi HTML. Possiamo utilizzarlo per applicare uno stile globale o per azioni specifiche su tutti gli elementi.

Ad esempio:

```
● ● ●  
* {  
    margin: 0;  
    padding: 0;  
}
```



Questo imposta il margine e il padding di tutti gli elementi a zero, eliminando gli spazi predefiniti del browser.

Selettore di classe

Le classi in CSS iniziano con un punto e possono essere assegnate a uno o più elementi HTML per applicare uno stile specifico.

Ad esempio:

```
● ● ●  
.titolo {  
    font-size: 24px;  
    color: #333333;  
}
```

Puoi assegnare questa classe a un elemento HTML in questo modo:

```
● ● ●  
<h1 class="titolo">Il mio Titolo</h1>
```

Questo applica uno stile personalizzato alla classe `titolo` definendo quindi un elemento HTML nella nostra pagina assegnando una classe e selezionando l'elemento nel file CSS grazie al selettore di classe `.titolo`. A differenza del selettore di tipo visto in precedenza, in questo caso assegniamo lo stile a tutti gli elementi HTML con classe `titolo` e non a tutti gli `h1`.

Possiamo avere all'interno della nostra pagina HTML diversi elementi con la stessa classe.



Selettore ID

Gli ID iniziano con un cancelletto # e possono essere utilizzati per selezionare uno specifico elemento HTML, devono essere **univoci all'interno della pagina**.

Ad esempio:

```
● ● ●  
#header {  
    background-color: #f2f2f2;  
    padding: 10px;  
}
```

Puoi assegnare questo ID a un elemento HTML in questo modo:

```
● ● ●  
<div id="header">Il mio Header</div>
```

Questo applica uno stile specifico all'elemento con l'ID header.
All'interno di una pagina HTML possiamo avere un solo elemento con lo stesso id ma più elementi con id diversi.



Selettore di raggruppamento

E' anche possibile raggruppare gli elementi in un solo selettore separando gli elementi da virgolette nel modo seguente:

```
● ● ●  
h1,  
.titolo {  
  font-size: 24px;  
  color: #333333;  
}
```

In questo modo puoi attribuire a tutti gli elementi `h1` e a tutti gli elementi con classe `.titolo` la grandezza del font di `24px` ed il colore grigio. Il selettore di raggruppamento è molto utile per non ripetere gli attributi CSS e raggruppare gli elementi con uno stile comune.

Selettori composti

I selettori composti ti permettono di specificare sempre più un elemento da selezionare.

```
● ● ●  
h1.titolo {  
  font-size: 24px;  
  color: #333333;  
}
```

attraverso `h1.titolo` potrai selezionare solo gli elementi `h1` che hanno anche classe `titolo`.



Selettore descendente

Il selettore descendente seleziona tutti gli elementi figli di un determinato elemento, a qualsiasi livello di annidamento.

```
● ● ●  
div p {  
    font-size: 24px;  
    color: #333333;  
}
```

Con questa regola di stile verranno selezionati tutti gli elementi `<p>` che sono discendenti di un elemento `<div>`.

Selettore di attributo

Il selettore di attributo permette di selezionare gli elementi che hanno un determinato attributo con un valore specifico.

```
● ● ●  
input[type="text"] {  
    font-size: 18px;  
    color: #666666;  
    background-color: #f7f7f7;  
}
```

Selezione tutti gli elementi `<input>` di tipo testo.



Esercizi

Crea una pagina HTML con un titolo, un paragrafo e un link.

Usa il selettore di tipo per cambiare il colore del testo del titolo in rosso e il selettore di classe per cambiare il colore del testo in verde.

Crea una pagina HTML con una lista non ordinata contenente tre elementi di lista. Usa il selettore di ID per cambiare il colore del testo del primo elemento di lista in arancione e il selettore di classe per cambiare il colore del secondo elemento di lista in viola.

Crea una pagina HTML con un campo di input di tipo testo, un campo di input di tipo password e un pulsante. Usa il selettore di tipo per cambiare il colore del testo dei campi di input in grigio, il selettore di classe per cambiare il colore del testo del pulsante in bianco e il selettore di attributo per cambiare il colore di sfondo dell'input di tipo password in blu.



Specificità

In CSS la specificità è un concetto che determina quale stile si applicherà a un elemento quando ci sono regole di stile conflittuali.

Analizzando il codice CSS seguente, di che colore sarà il testo del paragrafo?



```
<p id="special" class="highlight" style="color: purple;">  
    Ciao, sono un paragrafo!  
</p>
```



```
p {  
    color: blue;  
}  
  
.highlight {  
    color: red;  
}  
  
#special {  
    color: green;  
}
```

La specificità viene utilizzata per risolvere le collisioni tra le regole CSS e stabilire quale regola dovrebbe avere la precedenza su un'altra.

La specificità viene rappresentata da un numero e può essere calcolata in base ai selettori utilizzati. Un selettore più specifico avrà una specificità maggiore e, quindi, avrà la precedenza su un selettore meno specifico.

Il valore di specificità è spesso rappresentato da quattro segmenti numerici (ad esempio, 0-0-0-0), e più un segmento ha un valore alto, maggiore è la specificità. Quindi, quando ci sono conflitti tra le regole, l'elemento con la specificità più alta avrà la precedenza.



Ecco una suddivisione più dettagliata dei valori:

elementi e pseudo-elementi:

```
p {  
    color: blue;  
}  
specificità 0-0-0-1
```

classi, attributi e pseudo-classi:

```
.highlight {  
    color: red;  
}  
specificità 0-0-1-0
```

ID:

```
#special {  
    color: green;  
}  
specificità 0-1-0-0
```

stile inline:

```
<p id="special" class="highlight" style="color: purple;">  
    Ciao, sono un paragrafo!  
</p>  
specificità 1-0-0-0
```

In generale, è preferibile evitare l'uso eccessivo di ID per gli stili e favorire l'utilizzo di classi e altri selettori, poiché ID molto specifici possono portare a problemi di manutenzione e flessibilità nel lungo periodo. Come abbiamo visto in precedenza è, in generale, preferibile evitare anche l'uso eccessivo di stile inline.



Esercizi

Crea una pagina HTML con un titolo, un paragrafo e un link, segui il nome delle classi e degli ID come specificato di seguito e usa il metodo del CSS in file esterno e definisci le seguenti regole:

`h1` di colore rosso

`p` di colore verde

`link` di colore blu

Assegna al titolo `h1` una classe e un id titolo.

Per la classe `.titolo` assegna un colore rosa e per l'id `#titolo` assegna un colore giallo.

Collega il file cos alla pagina HTML e osserva il risultato. Quale colore avrà il titolo, il paragrafo e il link? Perché?



I commenti

I commenti in CSS servono per aggiungere note esplicative al codice e non hanno alcun effetto sul layout del documento.

La sintassi per i commenti in CSS è la seguente:

```
/* questo è un commento in CSS */
```

I commenti possono essere inseriti ovunque e possono essere usati su una singola riga o più righe.

I commenti sono utili per documentare il codice e facilitarne la comprensione e la manutenzione e per testare variazioni di codice senza eliminarlo o modificarlo definitivamente.

Usa i commenti in modo chiaro e coerente, commentare ti aiuterà anche a comprendere il tuo codice dopo molto tempo. Usa i commenti per spiegare il motivo di una scelta di codice o di un valore.



Unità di misura

In CSS, le unità di misura sono utilizzate per definire dimensioni e posizioni degli elementi all'interno di una pagina web. Le unità di misura possono essere categorizzate in unità relative e assolute. Le unità relative sono basate su proprietà dinamiche, come la dimensione del carattere del testo o la larghezza della finestra di visualizzazione, mentre le unità assolute sono fisse e non dipendono da fattori esterni.

Vediamo alcune unità di misura più utilizzate:

px - pixel - unità di misura assoluta, rappresenta un singolo punto sullo schermo. Esempio: `width: 100px;` imposta la larghezza di un elemento a 100px.

% - percentuale - la percentuale è una unità di misura relativa rispetto al contenitore genitore, utile per adattare le dimensioni agli elementi genitori, consentendo un layout flessibile e responsivo.

Esempio: `width: 50%;` imposta la larghezza a metà del contenitore genitore.

vw - viewport width - è un'unità relativa alla percentuale della larghezza della finestra di visualizzazione. Utile per adattare elementi in base alla larghezza della finestra del browser.

Esempio: `font-size: 5vw;` imposta la dimensione del testo al 5% della larghezza della finestra.

vh - viewport height - è un'unità relativa alla percentuale dell'altezza della finestra di visualizzazione. Ottimo per dimensionare gli elementi in base all'altezza della finestra del browser.

Esempio: `height: 80vh;` imposta l'altezza all'80% della finestra di visualizzazione.

rem room em: un'unità relativa alla dimensione del carattere del tag `<html>`. Utile per creare layout scalabili, mantenendo la coerenza in tutta la pagina.



Esempio: `font-size: 2rem;` imposta la dimensione del testo a 2 volte la dimensione del carattere radice.

`em`: unità relativa alla dimensione del carattere del suo elemento genitore.

Utile per dimensionare elementi in relazione alla dimensione del loro genitore.

Esempio: `font-size: 1.5em;` imposta la dimensione del testo a 1,5 volte la dimensione del carattere del genitore.



Colori

La gestione dei colori è fondamentale per personalizzare l'aspetto di una pagina web. Esistono diverse notazioni per rappresentare i colori, ciascuna con le proprie caratteristiche e flessibilità. La capacità di scegliere e combinare colori in modo efficace contribuisce significativamente alla progettazione di interfacce web accattivanti e usabili.

Vediamo alcune notazioni più comuni e come utilizzarle.

Colore Hexadecimal (HEX): il colore hex è rappresentato da un codice esadecimale di sei cifre che definisce la quantità di rosso, verde e blu. E' semplice e compatto ed è ottimo per rappresentare colori solidi. Esempio: #3498db rappresenta un colore blu intenso.

Colore RGB (Red, Green, Blue): RGB utilizza tre valori numerici per definire la quantità di rosso, verde e blu in un colore.

E' una notazione flessibile, permette di specificare i valori di rosso, verde e blu separatamente.

Esempio: rgb(52, 152, 219) rappresenta un colore blu intenso.

Colore RGBA (Red, Green, Blue, Alpha): Simile a RGB ma con un valore aggiuntivo per rappresentare l'opacità del colore (alpha). L'opacità varia da 0 (trasparente) a 1 (completamente opaco).

Utile per colori con trasparenza, ad esempio per sfondi semitrasparenti.

Esempio: rgba(52, 152, 219, 0.5) rappresenta un colore blu semitrasparente.

Vediamo alcuni esempi:

Assegniamo il colore del testo arancione e il colore di sfondo grigio chiaro agli elementi HTML con classe .elemento1 utilizzando la notazione hex.

```
● ● ●  
.elemento1 {  
    color: #ff5733;  
    background-color: #f2f2f2;  
}
```



Definiamo un bordo di 2px di colore rosso agli elementi con classe .elemento2 utilizzando la notazione rgb.

```
● ● ●  
.elemento2 {  
    border: 2px solid rgb(255, 99, 71);  
}
```

Settiamo un colore di sfondo verde semitrasparente utilizzando la notazione rgba e il colore bianco al testo utilizzando la notazione hex agli elementi html con classe .elemento3.

```
● ● ●  
.elemento3 {  
    color: #ffffff;  
    background-color: rgba(46, 204, 113, 0.7);  
}
```

In CSS è possibile anche utilizzare il colore per nomi predefiniti, come visto in precedenza, e corrispondono ai dei valori esadecimales di colori. Ad esempio il nome “red” corrisponde al valore “#FF0000”, che indica un colore rosso puro. Ci sono circa 140 nomi di colore supportati da tutti i browser moderni.

In questa guida useremo spesso, per comodità, la nomenclatura per nome ma l'utilizzo dei colori per nome in CSS ha alcuni svantaggi. Innanzitutto, i nomi di colore sono limitati e non coprono tutte le sfumature possibili, inoltre, i nomi di colore possono essere ambigui o soggettivi, e non riflettono esattamente la percezione visiva del colore. Infine, i nomi di colore possono variare leggermente tra i diversi browser, a seconda della calibrazione dello schermo e delle impostazioni di sistema.



Per questi motivi, è preferibile non utilizzare i colori per nome in CSS, ma usare invece dei sistemi più precisi e universali, come il codice esadecimale o il formato RGB. Questi sistemi permettono di specificare il colore in base ai suoi componenti cromatici e di avere un maggiore controllo e una maggiore varietà di colori.



Sfondi

Lo sfondo è una componente essenziale del design web e ci permette di gestire il colore, le immagini, i gradienti e altri effetti che contribuiscono all'esperienza visiva dell'utente.

La proprietà `background` in CSS è utilizzata per definire lo sfondo di un elemento HTML, fornendo un'ampia gamma di opzioni per personalizzare l'aspetto visivo di un sito web.

Colori di sfondo

La proprietà `background-color` imposta un colore di sfondo per un elemento. E' una scelta comune per definire il colore di uno o più elementi sulla pagina.

Vediamo alcuni esempi:

Definire uno colore di sfondo grigio per l'intero corpo della pagina.

```
● ● ●  
body {  
    background-color: #f5f5f5;  
}
```

Definire un colore di sfondo bianco per un determinato elemento.

```
● ● ●  
.contenitore {  
    background-color: #ffffff;  
}
```



Immagine di sfondo

La proprietà `background-image` consente di utilizzare un'immagine come sfondo. L'immagine può essere una texture, un motivo o un'immagine personalizzata.



```
header {  
    /* Immagine di sfondo per l'intestazione */  
    background-image: url('header-background.jpg');  
    /* Copre completamente l'elemento con l'immagine */  
    background-size: cover;  
}
```

Per definire un'immagine di sfondo possiamo specificare diverse proprietà, `background-repeat` che determina se e come un'immagine di sfondo dovrebbe essere ripetuta all'interno di un elemento.

Alcuni valori possibili sono:

- `repeat` (di default): L'immagine si ripete sia orizzontalmente che verticalmente fino a riempire completamente l'area di sfondo.
- `no-repeat`: L'immagine non viene ripetuta e appare solo una volta.
- `repeat-x`: L'immagine si ripete solo orizzontalmente.
- `repeat-y`: L'immagine si ripete solo verticalmente.

`background-position` che determina la posizione di un'immagine di sfondo all'interno del suo contenitore. Può accettare valori specifici in termini di coordinate o termini chiave:

Coordinate: le coordinate specificano la posizione orizzontale e verticale dell'immagine rispetto agli angoli del suo contenitore. I valori possono essere espressi in pixel o percentuali.





```
.elemento {  
    /* 20px da sinistra e 40px dall'alto */  
    background-position: 20px 40px;  
}
```



```
.elemento {  
    /* Posizione al centro orizzontale e al 25% dall'alto */  
    background-position: 50% 25%;  
}
```

Termini chiave: valori chiave come top, bottom, left, right, center consentono di posizionare l'immagine in modo più intuitivo.



```
.elemento {  
    /* Angolo in alto a destra */  
    background-position: right top;  
}
```



```
.elemento {  
    /* Centro orizzontale, angolo in basso */  
    background-position: center bottom;  
}
```

Personalizzare `background-repeat` e `background-position` consente di controllare come le immagini di sfondo vengono visualizzate nei tuoi elementi, aggiungendo un livello di dettaglio e precisione al design.



Esercizi

Crea una pagina HTML com un titolo, un paragrafo e un link. Usa il formato hex per cambiare il colore del testo del titolo in #FF0000 (rosso), il colore del testo del paragrafo in #00FF00 (verde) e il colore del testo del link in #0000FF (blu).

Crea una pagina HTML con una lista non ordinata contente tre elementi di lista. Usa il formato rgb per cambiare il colore del testo degli elementi lista in `rgb(0, 255, 255)` ciano.

Crea una pagina HTML con un campo di input di tipo testo, un campo di input di tipo password e un pulsante. Usa il formato roba per cambiare il colore del testo dei campi di input in `rgba(0, 0, 0, 0.5)` (nero semi-trasparente) e il colore del testo e dello sfondo del pulsante in `rgba(255, 255, 255, 1)` (bianco opaco) e `rgba(0, 0, 0, 0.8)` (nero quasi opaco).

Crea una pagina HTML con un titolo, un paragrafo e un link. Usa la proprietà `background-color` per cambiare il colore di sfondo del titolo in rosso, il colore di sfondo del paragrafo in verde e il colore di sfondo del link in blu.

Crea una pagina HTML con un `div` che contiene un campo di input di tipo testo, un capo di input di tipo password e un pulsante. Definisci un'immagine di sfondo per il div contenitore e usa la proprietà `background-repeat` per modificare il modo in cui le immagini di sfondo si ripetono negli elementi.



Font e testi

I font sono un elemento fondamentale per il design e la leggibilità di una pagina web. Con CSS, possiamo impostare il tipo, lo stile, il peso e la famiglia dei font da usare nei documenti HTML.

Per impostare il tipo di font, possiamo usare la proprietà CSS `font-family`, che accetta una lista di nomi di font separati da virgolette.

Questa lista serve a creare un sistema di fallback, in caso il primo font non sia disponibile sul dispositivo dell'utente. Se il font ha più di una parola nel nome, devi metterlo tra virgolette: “Times New Roman”. Puoi anche usare dei font generici, come `serif`, `sans-serif` o `monospace`, che corrispondono a dei font predefiniti dal browser ma solitamente useremo dei font specifici per ogni nostro progetto.

Per impostare la dimensione del font, usiamo la proprietà CSS `font-size`, che accetta diversi tipi di valore, come `px`, `em`, `rem` o `%`.

Per impostare lo stile del font useremo invece la proprietà `font-style`, che accetta valori `normal`, `italic` o `oblique`. Questa proprietà serve a rendere il font normale, corsivo o inclinato.

Useremo `font-weight` invece per impostare il peso del font ed accetta dei valori numerici da 100 a 900, oppure dei valori predefiniti come `normal`, `bold`, `lighter`, `bolder`, ecc... Questa proprietà serve a rendere il font più o meno spesso.

Per impostare la famiglia del font, usiamo la proprietà `font-family` che accetta una lista di nomi di famiglie di font separati da virgolette. Questa lista serve a raggruppare dei font simili tra loro, in modo da garantire una visualizzazione consistente su diversi sistemi o browser. Le famiglie di font più comuni sono `serif`, `sans-serif` e `monospace`, ma esistono anche altre categorie come `cursive`, `fantasy`, ...

Un tipo di font molto comune è il font `sans-serif`, che significa senza grazie. Le grazie sono quelle piccole linee che si trovano agli estremi delle lettere in alcuni font, come il `Times New Roman`. I font `sans-serif` hanno linee pulite e semplici, che creano un aspetto moderno e minimalista.



Useremo sia font serif, sia sans serif, a seconda delle esigenze e dalle caratteristiche grafiche del progetto.

Ad esempio, questo codice applica il font Arial a tutti i paragrafi della pagina. Se il font Arial non è disponibile, il browser userà un altro font sans-serif.

```
● ● ●  
p {  
    font-family: "Arial", sans-serif;  
}
```

Possiamo anche usare più nomi di font separati da virgole, per avere più opzioni di fallback. Per esempio:

```
● ● ●  
h1 {  
    font-family: "Verdana", "Helvetica", "Tahoma", sans-serif;  
}
```

Questo codice applica il font Verdana a tutti gli elementi h1 della pagina. Se il font Verdana non è disponibile, il browser proverà ad usare il font Helvetica, poi il font Tahoma, e infine un font sans-serif generico.

Uno strumento molto utile sono i Google Fonts che sono una collezione di font gratuiti e di alta qualità da utilizzare nei nostri progetti web.

Per usare un google font, devi prima scegliere il font che ti piace dal sito [Google Fonts](#), dove puoi anche vedere degli esempi di come appare il font e quali stili e varianti sono disponibili. Poi devi copiare il codice che ti viene fornito e incollarlo nel tuo file HTML, nella sezione <head>.



Per esempio:

```
● ● ●  
<head>  
  <link rel="preconnect" href="https://fonts.gstatic.com">  
  <link href="https://fonts.googleapis.com/css?  
family=Roboto:300,400,700&display=swap" rel="stylesheet">  
</head>
```

Questo codice carica il font `Roboto` con i pesi 300, 400 e 700. Puoi scegliere i pesi che ti servono in base al tuo design. Infine, devi applicare il font ai tuoi elementi utilizzando la proprietà `font-family` in CSS.

Per esempio:

```
● ● ●  
body {  
  font-family: "Roboto", sans-serif;  
}
```

Questo codice applica il font `Roboto` a tutto il corpo del documento HTML. Puoi anche applicare il font solo ad alcuni elementi, come i titoli o i paragrafi.

Ricordati di usare sempre un font di riserva (fallback) nel caso in cui il Google Font non si carichi correttamente. In questo esempio, il font di riserva è `sans-serif`, che indica al browser di usare un font generico senza grazie.

I vantaggi di usare i Google font sono:

- sono gratuiti e di qualità
- sono ottimizzati per il web e hanno una buona compatibilità con i browser
- sono facili da integrare e da personalizzare
- offrono una vasta di scelta di stili e di categorie
- migliorano l'aspetto e l'identità del tuo sito web



In generale utilizziamo un font personalizzato quando vogliamo creare un design unico e distintivo, per trasmettere una certa personalità e per migliorare l'usabilità e l'accessibilità dei siti web.

Oltre alla gestione della famiglia del font possiamo impostare diverse proprietà per modificare e migliorare l'aspetto visivo dei testi.

Ad esempio, come visto in precedenza, la proprietà `text-align` serve per allineare il testo all'interno di un elemento a blocchi. I valori possibili sono, `left`, `right`, `center` e `justify`.

La proprietà `text-decoration` serve per aggiungere degli effetti visivi al testo, come sottolineature, linee sopra o attraverso il testo, o lampeggiamenti. I valori possibili sono `none`, `underline`, `overline`, `line-through` e `blink`.

La proprietà `text-transform`, invece, serve per modificare il formato del testo, trasformandolo in maiuscolo, minuscolo o capitalizzando la prima lettera di ogni parola. I valori possibili sono `none`, `uppercase`, `lowercase` e `capitalize`.

Le proprietà `letter-spacing` e `word-spacing` servono per aumentare o diminuire la distanza tra le lettere o le parole del testo.



Link

I link sono elementi fondamentali per il web, perché permettono di navigare tra le pagine e di accedere a contenuti di interesse. Personalizzare i link con CSS significa rendere più attraente e funzionale un sito web, oltre che a migliorare l'esperienza utente e l'accessibilità.

Le proprietà che puoi usare per gestire i link sono molte, come abbiamo visto possiamo modificare il colore, la decorazione, il background, la famiglia e il peso del font. Oltre a queste proprietà, possiamo usare dei selettori speciali per applicare stili diversi ai link a seconda del loro stato. I quattro stati possibili sono:

- `a:link` indica un link normale, non visitato
- `a:visited` indica un link che l'utente ha già visitato
- `a:hover` indica un link quando l'utente passa il mouse sopra
- `a:active` indica un link nel momento in cui viene cliccato

Vediamo ora alcuni esempi di come usare queste proprietà e questi selettori per personalizzare i link.

Vogliamo creare dei link con un colore blu, senza sottolineatura, e con un effetto di ingrandimento quando si passa il mouse sopra.

```
● ● ●  
  
a {  
    color: blue;  
    text-decoration: none;  
}  
  
a:hover {  
    font-size: 120%;  
}
```



Adesso vogliamo creare dei link di un colore rosso, con una sottolineatura tratteggiata e con un cambio di colore e di sfondo quando si passa il mouse sopra.

```
● ● ● ● ●  
a {  
    color: red;  
    text-decoration: underline;  
    text-decoration-style: dashed;  
}  
  
a:hover {  
    color: green;  
    background-color: blue;  
}
```



Esercizi

Crea una pagina HTML con un titolo, un paragrafo e un link. Usa la proprietà `font-family` per cambiare il font del titolo in `Arial`, il font del paragrafo in `Times New Roman` e il font del link in `Courier New`.

Crea una pagina HTML con una lista non ordinata contenente tre elementi di lista. Usa la proprietà `font-size` per cambiare la dimensione degli elementi a `1.5em`.

Crea una pagina HTML con un campo di input di tipo testo, un campo di input di tipo password e un pulsante. Usa la proprietà `font-weight` per cambiare il peso del testo dei campi di input in `normal`, il peso del testo del pulsante quando si passa sopra con il mouse in `bold`.

Crea una pagina HTML con un titolo, un paragrafo e un link. Usa la proprietà `text-align` per cambiare l'allineamento del testo del titolo in `center`, l'allineamento del testo del paragrafo in `justify` e l'allineamento del testo del link in `right`.

Crea una pagina HTML con tre paragrafi di testo, usa la proprietà `text-decoration` per cambiare la decorazione del testo del primo elemento in `underline`, la decorazione del testo del secondo paragrafo in `overline` e del terzo paragrafo in `line-through`.

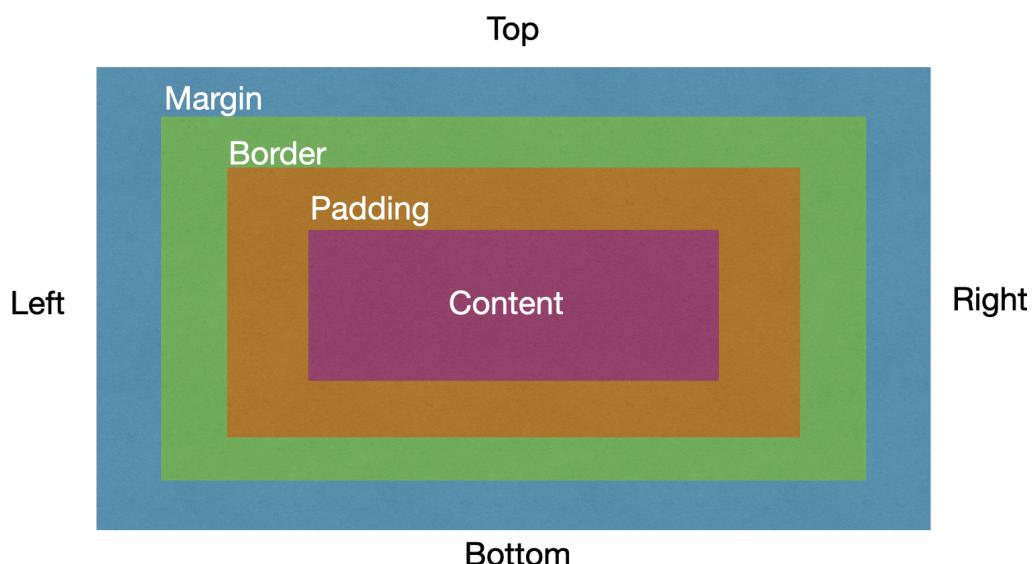
Crea una pagina HTML con un campo di input di tipo testo, un campo di input di tipo password e un pulsante. Usa la proprietà `text-transform` per cambiare la trasformazione del testo dei campi input in `uppercase`, la trasformazione del testo del pulsante in `lowercase` e la trasformazione del testo del pulsante quando si passa sopra con il mouse in `capitalize`.

Crea una pagina HTML con un titolo, un paragrafo e un link. Usa la proprietà `a:hover` per cambiare il colore del testo del link in rosso quando il mouse passa sopra al link.



Box Model

Il Box Model in CSS è un concetto fondamentale per capire come vengono visualizzati gli elementi HTML sulla pagina. Il Box Model descrive la struttura di ogni elemento come una scatola che ha quattro componenti: contenuto, padding, bordo e margine. Ogni componente ha una dimensione e una posizione che influenzano lo spazio occupato dall'elemento e la sua relazione con gli altri elementi.



Il contenuto è la parte interna della scatola, dove si trovano il testo e le immagini dell'elemento. La dimensione del contenuto è determinata dalle proprietà CSS `width` e `height`.

Le proprietà `width` e `height` impostano la larghezza e l'altezza di un elemento HTML, rispettivamente. Queste proprietà si riferiscono solo all'area dei contenuti dell'elemento. Il valore di `width` e `height` può essere espresso in unità di misura assolute (come `px`) o relative come (come `%`, `em`, `rem`, ...). Se non si specifica nessun valore, il browser calcolerà automaticamente le dimensioni in base al tipo di elemento e al suo contenuto. Puoi anche usare le proprietà `min-width`, `max-width`, `min-height` e `max-height` per definire una dimensione minima o massima per l'elemento.



Il `border` serve a creare un bordo intorno all'elemento ed è una scorciatoia che combina tre sotto proprietà: `border-width`, `border-style` e `border-color`. Queste sotto proprietà servono per impostare lo spessore, lo stile e il colore del bordo, rispettivamente. Il valore di `border-width` può essere espresso in unità di misura assolute o relative e il valore di `border-style` può essere uno tra i seguenti: `none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`. Il valore di `border-color` può essere espresso con un nome di colore, un valore esadecimale, una funzione `rgb()` o `rgba()`, o con la parola chiave `currentcolor`.

`margin` e `padding` servono per creare dello spazio interno a un elemento HTML. La differenza tra i due è che il `margin` si riferisce allo spazio esterno all'elemento, mentre il `padding` si riferisce allo spazio interno all'elemento. Il valore di `margin` e `padding` può essere espresso in unità di misura assolute o relative. Puoi anche usare le proprietà `margin-top`, `margin-right`, `margin-bottom` e `margin-left` per modificare solo un lato dell'elemento e analogamente per il `padding` con `padding-top`, `padding-right`, `padding-bottom` e `padding-left`.

Per calcolare la dimensione totale di un elemento, bisogna sommare le dimensioni di tutti e quattro i componenti. Ad esempio, se un elemento ha una larghezza di contenuto di 200px, un padding di 10px, un bordo di 5px e un margine di 20px, la sua larghezza totale sarà:

$$200 + 2 \times (10 + 5 + 20) = 270\text{px}$$



Per rappresentare un elemento con classe `elemento1` con una larghezza predefinita di 200px con spazio interno di 20px, un bordo di 5px solido e di colore nero e uno spazio con gli altri elementi di 10px scriveremo:

```
● ● ●  
.elemento1 {  
    width: 300px;  
    padding: 20px;  
    border: 5px solid black;  
    margin: 10px;  
}
```

In quest'altro esempio rappresentiamo un elemento `p` di larghezza 50% del suo contenitore genitore e con padding, border e margin espressi in em:

```
● ● ●  
p {  
    width: 50%;  
    padding: 1em;  
    border: 0.5em dashed red;  
    margin: 2em;  

```

Ancora nel seguente esempio vediamo l'utilizzo di una nuova proprietà `box-sizing`:

```
● ● ●  
.elemento2 {  
    box-sizing: border-box;  
    width: 200px;  
    padding: 10px;  
    border: 10px solid blue;  
    margin: 5px;  
}
```



Il `box-sizing` in CSS è una proprietà che determina come vengono calcolate le dimensioni di un elemento HTML, includendo o escludendo il padding e il bordo. Ci sono due valori possibili per il `box-sizing`: `content-box` e `border-box`.

`content-box` è il valore predefinito e segue il modello standard dei CSS. In questo caso, le dimensioni di un elemento sono date solo dall'area dei contenuti, mentre il padding e il border sono aggiunti in più. Questo significa che se imposti la larghezza e l'altezza di un elemento a 100px, ma aggiungi anche un padding di 10px e un border di 5px, l'elemento avrà una dimensione effettiva di 130px.

`border-box` è il valore alternativo e modifica il modello standard CSS. In questo caso, le dimensioni di un elemento includono anche il padding e il border, mentre l'area dei contenuti si riduce di conseguenza. Questo significa che se imposti la larghezza e l'altezza di un elemento a 100px, ma aggiungi anche un padding di 10px e un border di 5px, l'elemento avrà una dimensione effettiva di 100px.

Il `box-sizing` è una proprietà utile per gestire meglio lo spazio occupato degli elementi e il loro posizionamento.

Con i concetti del box model possiamo rappresentare gli elementi all'interno della pagina HTML con molta precisione e definire le dimensioni di ogni singolo elemento.

Vediamo, con qualche esempio, come è possibile utilizzare per `margin` e `padding` diverse notazioni che ci permetteranno di avere pieno controllo del codice CSS e degli elementi in pagina.

CSS ci mette a disposizione una notazione compatta e una estesa per definire al meglio `margin` e `padding`. Nella notazione compatta si usa una sola proprietà per impostare il padding e il margine di tutti e quattro i lati dell'elemento. Il valore può essere uno, due, tre o quattro, a seconda di quanti lati si vuole modificare. Ecco il significato di ogni caso:

- Un valore: si applica a tutti i lati. Ad esempio, `padding: 10px` significa che il padding è di 10px per ogni lato.



- Due valori: il primo si applica al lato superiore e inferiore, il secondo al lato destro e sinistro. Ad esempio, `margin: 20px 40px;` significa che il margine è di `20px` per il lato superiore e inferiore, e di `40px` per il lato destro e sinistro.
- Tre valori: il primo si applica al lato superiore, il secondo al lato destro e sinistro, il terzo al lato inferiore. Ad esempio, `padding: 10px 20px 30px;` significa che il padding è di `10px` per il lato superiore, di `20px` per il lato destro e sinistro e `30px` per il lato inferiore.
- quattro valori: si applicano in senso orario, partendo dal lato superiore. Ad esempio, `margin: 10px 20px 30px 40px;` significa che il margine è di `10px` per il lato superiore, di `20px` per il lato destro, di `30px` per il lato inferiore e di `40px` per il lato sinistro.

Nella nozione estesa si usano quattro proprietà separate per impostare il padding o il margine di ogni lato dell'elemento. Il valore può essere uno solo per ogni proprietà, per il margine quindi `margin-top`, `margin-right`, `margin-bottom` e `margin-left` e per il padding `padding-top`, `padding-right`, `padding-bottom` e `padding-left`.



Esercizi

Crea una pagina HTML con un titolo, un paragrafo e un link. Usa le proprietà width, height, margin, border e padding per impostare le dimensioni e lo spazio di ogni elemento come segue:

Titolo con larghezza 300px, altezza 50px, margine superiore 10px, bordo solido di colore nero e spessore 5px.

Paragrafo con larghezza 200px, altezza 100px, margine di 20px, sfondo di colore giallo e un padding di 5px;

Link con larghezza di 100px, altezza di 20px, margine superiore e inferiore di 10px e sinistro e destro di 20px, padding superiore di 20px e padding inferiore di 30px;

Crea una pagina HTML con una lista non ordinata contenente tre elementi di lista. Usa la proprietà box-sizing per modificare il modo in cui il box model calcola le dimensioni e il bordo degli elementi.

La lista avrà una larghezza di 400px, un'altezza di 200px, un margine di 0, un bordo di 5px solido verde e un padding di 10px.

Crea un div contenitore che racchiuda un titolo, un paragrafo e un link.

Il contenitore avrà una larghezza massima di 400px e una larghezza di 100%.

Imposta un margine superiore di 20px e un margine sinistro di 40px e uno sfondo di colore blu.

Adesso prova a sostituire il valore della proprietà margin in 0 auto, in questo modo:

```
margin: 0 auto;
```

Dove si posizionerà il div contenitore rispetto alla finestra?

Crea una pagina HTML con un elemento di blocco che contiene due elementi inline: uno span e un link. Assegna a ciascun elemento delle dimensioni, un padding, un bordo e un margine diversi. Osserva come si comportano gli elementi inline rispetto al box model e come si dispongono orizzontalmente.



Crea una pagina HTML con un'immagine all'interno. Usa il CSS per impostare la larghezza dell'immagine a 200px, il padding a 15px, il bordo a 10px e il margine a 25px. Osserva come cambia la dimensione totale dell'immagine in base al valore del `box-sizing`.

Crea una pagina HTML con un paragrafo all'interno. Usa il CSS per aggiungere un bordo arrotondato di colore verde di 2px al paragrafo. Osserva come il bordo arrotondato si adatta al padding del contenuto del paragrafo.

Crea una pagina HTML con un titolo e un sottotitolo all'interno. Usa il CSS per impostare l'altezza del titolo a 100px, il padding a 20px, il bordo a 10px e il margine a 30px. Usa il CSS per impostare l'altezza del sottotitolo a 50px, il padding a 10px, il bordo a 5px e il margine a 15px. Osserva come cambia l'altezza totale del titolo e del sottotitolo in base al valore del `box-sizing`.



Posizioni

Il posizionamento in CSS è il modo di controllare la disposizione degli elementi HTML sulla pagina web. E' utile per creare layout responsivi e flessibili, oltre che per gestire gli effetti visivi come sovrapposizioni, menu fissi, animazioni, ecc...

La proprietà principale per il posizionamento è `position`, che può assumere valori diversi:

- `static`
- `relative`
- `absolute`
- `fixed`
- `sticky`
- ...

Ogni valore determina un diverso comportamento dell'elemento rispetto al suo contenitore, al flusso del documento e alla finestra del browser. Per spostare un elemento, dalla sua posizione di default, possiamo usare le proprietà `top`, `right`, `bottom` e `left` e la proprietà `z-index` per regolare l'ordine di sovrapposizione degli elementi.

Vediamo nel dettaglio i diversi valori di `position` e quando è preferibile usare un valore piuttosto che un altro.



position: static

è il valore di default per tutti gli elementi, significa che l'elemento segue il normale flusso del documento e non si può spostare con le proprietà top, right, bottom e left. Viene usato, di default, quando non si vuole modificare la posizione naturale di un elemento.

Ad esempio:



```
<div class="box-orange"></div>
<div class="box-blue"></div>
```



```
.box-orange {
    background: orange;
    height: 100px;
    width: 100px;
}

.box-blue {
    background: blue;
    height: 100px;
    width: 100px;
    position: static;
}
```

Con questo codice rappresentiamo due quadrati colorati uno sotto l'altro, senza alcuno spostamento perché nel primo caso, box-orange, non specifichiamo altri valori di position e di default è static mentre nel secondo caso specifichiamo static che non modifica la posizione dell'elemento.



position: relative

Significa che l'elemento viene posizionato relativamente alla sua posizione normale nel flusso del documento.

Si possono usare le proprietà `top`, `right`, `bottom` e `left` per spostare l'elemento da una posizione di partenza, ma senza influenzare la posizione degli altri elementi.

Si usa quando si vuole creare un effetto di traslazione o quando si vuole creare un contenitore per un elemento posizionato in modo assoluto.

Ad esempio:

```
<div class="box-orange"></div>
<div class="box-blue"></div>
```

```
.box-orange {
    background: orange;
    height: 100px;
    width: 100px;
}

.box-blue {
    background: blue;
    height: 100px;
    width: 100px;
    position: relative;
    top: 50px;
    left: 50px;
}
```

Questo codice rappresenta due quadrati colorati, ma il secondo è spostato di `50px` verso il basso e `50px` verso destra rispetto alla sua posizione normale.

Il primo quadrato non è influenzato dallo spostamento del secondo.



position: absolute

significa che l'elemento viene posizionato rispetto al suo contenitore più vicino che ha una posizione diversa da `static`. Se non c'è nessun elemento contenitore con una posizione diversa da `static`, l'elemento viene posizionato rispetto al `body` del documento.

Si possono usare le proprietà `top`, `right`, `bottom` e `left` per spostare l'elemento da una posizione di partenza, che dipende dal contenitore di riferimento. Si usa quando si vuole creare un effetto di sovrapposizione o quando si vuole posizionare un elemento in un punto preciso della pagina.

Ad esempio:

```
<div class="contenitore">
    <div class="box-orange"></div>
    <div class="box-blue"></div>
</div>
```

```
.contenitore {
    position: relative;
    width: 800px;
    height: 400px;
}

.box-orange {
    position: relative;
    background: orange;
    height: 200px;
    width: 200px;
}

.box-blue {
    position: absolute;
    background: blue;
    height: 100px;
    width: 100px;
    top: 50px;
    right: 50px;
}
```



Con questo codice si rappresentano due quadrati colorati, ma il secondo è posizionato in modo assoluto rispetto al suo contenitore. Il secondo quadrato è spostato di 50px verso il basso e 50px dal lato destro dell'elemento contenitore. Se al contenitore non avessimo dato un valore di position relative il box-blue si sarebbe posizionato a 50px dal bordo destro della pagina.

position: fixed

Significa che l'elemento viene posizionato rispetto alla finestra del browser e non si muove quando si fa lo scroll della pagina. Si possono usare le proprietà top, right, bottom e left per spostare l'elemento da una posizione di partenza, che dipende dalla finestra del browser. Si usa quando si vuole creare un elemento che rimane sempre visibile, come un menu fisso o un pulsante di ritorno in alto.

Ad esempio:

```
● ● ●  
<div class="box-orange"></div>  
<div class="box-blue"></div>
```

```
● ● ●  
.box-orange {  
    background: orange;  
    height: 100px;  
    width: 100px;  
}  
  
.box-blue {  
    background: blue;  
    height: 100px;  
    width: 100px;  
    position: fixed;  
    bottom: 50px;  
    left: 50px;  
}
```



In questo esempio il box blu è posizionato in modo fisso rispetto alla finestra del browser ed è spostato 50px da sinistra e 50px dal basso e rimane sempre in quella posizione anche quando si fa lo scroll della pagina.

position: sticky

Significa che l'elemento si comporta come static fino a quando non raggiunge un limite impostato con top, right, bottom o left, dopodiché si comporta come fixed.

L'elemento rimane nel flusso del documento e occupa spazio. Si usa quando si vuole creare un elemento che rimane visibile fino a un certo punto, un menu o l'intestazione di una tabella e in tantissimi altri casi.

Ad esempio:

```
<div class="box-orange"></div>
<div class="box-blue"></div>
```

```
.box-orange {
    background: orange;
    height: 100px;
    width: 100px;
}

.box-blue {
    position: sticky;
    background: blue;
    height: 100px;
    width: 100px;
    top: 0;
}
```



Questo codice produce due quadrati colorati, ma il secondo è posizionato in modo `sticky` rispetto alla finestra del browser. Il quadrato blu segue il normale flusso del documento fino a quando non si raggiunge il bordo superiore della finestra, dopodiché rimane fisso in quella posizione anche quando si fa lo scroll della pagina.

Un'altra proprietà molto utile del posizionamento è `z-index` che specifica l'ordine di sovrapposizione di un elemento posizionato e dei suoi discendenti. Gli elementi con `z-index` maggiore coprono quelli con un valore minore. `z-index` funziona solo sugli elementi posizionati in `relative`, `absolute`, `fixed` e `sticky` e sugli elementi `flex` che vedremo più avanti.

Il valore di `z-index` può essere `auto` o un numero intero. Il valore `auto` significa che l'elemento non stabilisce un nuovo contesto di impilamento locale e il suo ordine di impilamento è uguale a quello dei suoi genitori. Il valore intero significa che l'elemento stabilisce un contesto di impilamento locale e il suo ordine è determinato dal numero. Anche i numeri negativi sono consentiti, ma presta attenzione.

Per esempio se hai due elementi posizionati assolutamente che si sovrappongono, puoi usare `z-index` per controllare quale elemento appare sopra l'altro. Se l'elemento A ha `z-index: 1;` e l'elemento B ha `z-index: 2,` l'elemento B apparirà sopra l'elemento A.

Queste sono le principali modalità di posizionamento in CSS, ma ci sono molte altre cose da sapere per gestire al meglio la disposizione degli elementi, impareremo passo dopo passo ad utilizzare al meglio queste proprietà.

Di seguito troverai alcuni esercizi utili per il posizionamento CSS. Per ogni esercizio dovrai creare una pagina HTML con un `div` all'interno e usare il CSS per impostare la posizione del `div` secondo le indicazioni. Puoi usare un editor online [CodePen](#) per provare il codice.



Esercizi

Imposta la posizione di un `div` come `relative` e spostalo di `50px` verso il basso e `20px` verso destra rispetto alla sua posizione originale.

Imposta la posizione di un `div` come `absolute` e posizionalo rispetto al bordo superiore destro della finestra del browser con una distanza di `30px`.

Imposta la posizione di un `div` a `fixed` e posizionalo rispetto al bordo inferiore destro della finestra del browser con una distanza di `20px`.

Consiglio, se non riesci a verificare lo scroll della pagina prova a racchiudere l'elemento box all'interno di un contenitore, imposta l'altezza del contenitore con un valore che superi la dimensione dell'altezza della pagina.

Imposta la posizione di un `div` come `sticky` e posizionalo rispetto al bordo superiore della finestra del browser in modo che segua lo scorrimento verticale.

Usando anche la proprietà `z-index`, per gestire la sovrapposizione degli elementi, prova a creare una pagina HTML con tre `div` all'interno, e usare il CSS per impostare la posizione e lo `z-index` dei `div` secondo le indicazioni:

- il primo `div` con `position: relative;` e `z-index: 1;` e sia spostato di `50px` verso il basso e `50px` verso destra rispetto alla sua posizione normale.
- il secondo `div` con `position: absolute` e `z-index: 2`; e posizionato in alto a sinistra rispetto al primo `div`, con una distanza di `10px`.
- il terzo `div` con `position: fixed`; e `z-index: 3`; e posizionato in basso a destra rispetto alla finestra del browser, con una distanza di `20px`.



Display

Il `display` in CSS è una proprietà che serve a specificare come un elemento è mostrato sulla pagina web. Ogni elemento HTML ha un valore di `display` predefinito, a seconda del tipo di elemento che è. Il valore di `display` può influenzare il comportamento dell'elemento e il layout dei suoi contenuti.

Ci sono diversi tipi di attributi che si possono usare per il `display` in CSS, i più comuni sono:

block – l'elemento genera una scatola a blocco, che occupa tutta la larghezza disponibile e crea una nuova riga prima e dopo di sé. Esempi di elementi `block` sono `<p>`, `<div>`, `<h1>` e così via.

```
● ● ●  
p {  
    display: block;  
    background-color: red;  
}
```

inline – l'elemento genera una o più scatole in linea, che non creano una nuova riga prima e dopo di sé. In un flusso normale, l'elemento successivo sarà sulla stessa riga se c'è spazio.

Esempi di elementi `inline` sono ``, `<a>`, `` e così via.

```
● ● ●  
a {  
    display: inline;  
    color: blue;  
}
```



inline-block - l'elemento genera una scatola a blocco in linea, che si comporta come un elemento inline, ma a cui si possono applicare le proprietà di altezza e larghezza. Esempi di elementi inline-block sono <button>, <input> e così via.

```
● ● ●  
button {  
    display: inline-block;  
    width: 100px;  
    height: 50px;  
    background-color: green;  
}
```

none - fa scomparire completamente l'elemento senza lasciare spazio nel layout. Questo valore è adatto per elementi che devono essere nascosti in determinate situazioni.

```
● ● ●  
.modal {  
    display: none;  
}
```

Utilizzando la proprietà css `display` quindi possiamo definire ogni come viene mostrato ogni singolo evento, se assegneremo ad un div il valore `display: inline` si comporterà come uno , se invece assegniamo `display: block` ad un elemento a si comporterà come un <div>, e così via.



Visibility e Opacity

La proprietà `visibility` determina se un elemento è visibile o no. Se imposta `visibility: hidden`, l'elemento diventa invisibile, ma occupa ancora lo spazio nella pagina. Se imposta `visibility: visible`, l'elemento è visibile normalmente.

La proprietà `opacity` invece specifica il livello di trasparenza di un elemento, cioè il grado in cui il contenuto dietro l'elemento è visibile. Il valore di `opacity` va da 0 a 1, dove 0 è completamente trasparente e 1 è completamente opaco. Se imposta `opacity: 0`, l'elemento è invisibile, ma occupa ancora lo spazio nella pagina. Se imposta `opacity: 0.5`, l'elemento è semi-trasparente. Se imposta `opacity: 1`, l'elemento è opaco. La proprietà `opacity` si applica anche a tutti gli elementi figli dell'elemento. Diversamente da come abbiamo visto per il `display: none`; queste due proprietà mantengono comunque lo spazio dell'elemento all'interno della pagina.



Esercizi

Crea una pagina HTML con tre `<div>` di colore rosso, verde e blu. Imposta la proprietà `display` del primo `<div>` come `block`, del secondo come `inline` e del terzo come `none` e osserva il risultato.

Crea una pagina con tre elementi `` di colore rosso, verde e blu. Imposta la proprietà `display` per il primo `` come `block` e osserva il risultato confrontando.

Crea una pagina HTML con tre `<div>` di colore rosso, verde e blu. Imposta la proprietà `visibility` del secondo `<div>` a `hidden` e osserva il risultato.

Crea una pagina HTML con tre `<div>` di colore rosso, verde e blu. Imposta la proprietà del primo `<div>` a `0`, del secondo a `0.5` e del terzo a `1`, osserva il risultato e prova a variare l'`opacity` dei tre elementi con altri valori.



Liste

Le liste in HTML sono elementi che permettono di organizzare i contenuti in modo ordinato (``) e non ordinato (``). Per personalizzare l'aspetto delle liste, possiamo usare alcune proprietà CSS specifiche, come:

- `list-style-type`: definisce il tipo di marcatore per gli elementi della lista. Può avere un valore predefinito (come `circle`, `square`, `upper-roman`, ecc...) o `none` per rimuovere i marcatori.
- `list-style-image`: sostituisce il marcatore con un'immagine. Può essere un URL o `none` per annullare l'effetto.
- `list-style-position`: determina la posizione del marcatore rispetto al testo dell'elemento. Può essere `inside` (il marcatore fa parte del testo) o `outside` (il marcatore è separato dal testo).
- `list-style`: è una proprietà abbreviata che combina le tre precedenti in un'unica dichiarazione.

Ecco alcuni esempi di come usare queste proprietà per personalizzare le liste.

Cambiare il tipo di marcatore per una lista non ordinata.

```
● ● ●  
ul {  
    list-style-type: square;  
}
```

Usare un'immagine come marcatore per una lista ordinata.

```
● ● ●  
ol {  
    list-style-image: url("marcatore.png");  
}
```



Spostare il marcatore all'interno del testo per un elemento della lista.



```
li {  
    list-style-position: inside;  
}
```



Esercizi

Crea una lista non ordinata con i nomi dei tuoi animali preferiti. Cambia il marcatore in `circle`.

Crea una lista ordinata con i titoli dei tuoi libri preferiti. Usa un tipo di marcatore diverso da quello di default e allinea il testo a destra.

Crea una lista annidata con le annidata con le categorie e le sottocategorie di un negozio online. Usa un marcatore personalizzato per la categoria principale e un marcatore predefinito per la sottocategoria.



Pseudo Classi

Le pseudo classi in CSS sono dei selettori speciali che permettono di applicare gli stili agli elementi HTML in base al loro stato o alla loro posizione nel documento. Per esempio, puoi usare le pseudo classi per cambiare il colore dei link quando sono visitati o quando ci passi sopra con il mouse, per evidenziare il primo o l'ultimo elemento di una lista, per modificare lo stile di un elemento quando ricevi il focus o quando è selezionato e in tante altre situazioni.

La sintassi delle pseudo classi è la seguente:

```
selettore:pseudo-classe {  
    proprietà: valore;  
}
```

Il nome della pseudo classe è preceduto da due punti (:) e segue senza spazi il nome del selettore. Puoi usare le pseudo classi con tutti i tipi di selettori, come elementi, classi, id o attributi.

Avevamo visto precedentemente la gestione dei link, come applicare uno stile ai link già visitati, al passaggio del mouse, ecc... adesso vedremo le pseudo classi per gli elementi figli.

Gli elementi figli sono quelli che sono contenuti all'interno di un altro elemento, detto elemento padre. Per esempio, gli elementi `` sono figli dell'elemento `` o ``. Le pseudo classi per elementi figli ti permettono di selezionare e modificare lo stile di alcuni elementi figli in base alla loro posizione o al loro tipo. Queste sono le pseudo classi per gli elementi figli:

- `:first-child` si applica al primo elemento figlio di un elemento padre. Per esempio, puoi usare questa pseudo classe per cambiare il colore del primo elemento di una lista o per evidenziare il primo paragrafo di un testo.
- `:last-child` si applica all'ultimo elemento figlio di un elemento padre. Per esempio, puoi usare questa pseudo classe per aggiungere un bordo all'ultimo elemento di una lista.



- `:nth-child(n)` si applica all'elemento figlio che si trova nella posizione `n` rispetto al suo elemento padre. Il valore di `n` può essere un numero, una formula o le parole chiave `even` (pari) o `odd` (dispari). Per esempio, puoi usare questa pseudo classe per alternare il colore degli elementi di una lista.
- `:nth-last-child(n)` si applica all'elemento figlio che si trova nella posizione `n` rispetto alla fine del suo elemento padre. Il valore `n` segue le stesse regole di `:nth-child(n)`. Per esempio, puoi usare questa pseudo classe per evidenziare gli ultimi tre elementi di una lista.
- `:only-child` si applica all'elemento figlio che è l'unico figlio del suo elemento padre. Per esempio, puoi usare questa pseudo classe per nascondere il marcitore di una lista che ha un solo elemento.
- `:first-of-type` si applica al primo elemento figlio di un certo tipo di un elemento padre. Per esempio, puoi usare questa pseudo classe per cambiare il colore del primo elemento `<p>` all'interno di un elemento `<div>`.
- `:last-of-type` si applica all'ultimo elemento figlio di un certo tipo di un elemento padre. Per esempio, puoi usare questa pseudo classe per aggiungere un bordo all'ultimo elemento `<p>` all'interno di un elemento `<div>`.
- `:nth-child-type(n)` si applica all'elemento figlio di un certo tipo che si trova nella posizione `n` rispetto al suo elemento padre. Il valore di `n` segue le stesse regole di `:nth-child(n)`. Per esempio, puoi usare questa pseudo classe per alternare il colore degli elementi `<p>` all'interno di un elemento `<div>`.
- `:nth-last-of-type(n)` si applica all'elemento figlio di un certo tipo che si trova nella posizione `n` rispetto alla fine del suo elemento padre. Il valore di `n` segue le stesse regole di `:nth-child(n)`. Per esempio, puoi usare questa pseudo classe per evidenziare gli ultimi tre elementi `<p>` all'interno di un elemento `<div>`.
- `:only-of-type` si applica all'elemento figlio di un certo tipo che è l'unico di quel tipo del suo elemento padre. Per esempio, puoi usare questa pseudo classe per nascondere il marcitore di una lista che ha solo elementi ``.



Le pseudo classi per gli elementi figli si basano sulla posizione o sul tipo degli elementi, non sul loro contenuto o sul loro valore. Inoltre, le pseudo classi per gli elementi figli si applicano solo agli elementi che hanno un elemento padre.

Ecco alcuni esempi di come usare queste pseudo classi per personalizzare gli elementi figli:

Primo elemento figlio.



```
li:first-child {  
    color: blue;  
}
```

Ultimo elemento figlio.



```
li:last-child {  
    border-bottom: 1px solid black;  
}
```

Elemento figlio in posizione pari.



```
li:nth-child(even) {  
    background-color: lightgray;  
}
```



Elemento figlio in posizione dispari



```
li:nth-child(odd) {  
    background-color: red;  
}
```

Ultimi tre elementi figli.



```
li:nth-last-child(-n+3) {  
    font-weight: bold;  
}
```

Unico elemento figlio.



```
li:only-child {  
    list-style: none;  
}
```

Primo elemento figlio di tipo <p> .



```
p:last-of-type {  
    border-bottom: 1px solid black;  
}
```



Esercizi

Crea una pagina HTML con una lista non ordinata contenente cinque elementi di lista. Cambia il colore del testo del primo elemento di lista in rosso e il colore del testo dell'ultimo elemento in blu. Cambia anche il colore del terzo elemento della lista in verde.

Crea una pagina HTML con una lista ordinata contenente dieci elementi di lista. Cambia il colore di sfondo del primo elemento di lista in giallo e il colore di sfondo degli elementi pari in rosa.

Crea una pagina HTML con un paragrafo e tre elementi span. Cambia il colore del primo elemento in viola e del secondo elemento in grigio.

Crea una pagina HTML con un div che contenga cinque immagini. Cambia la dimensione della prima immagine in 200x200 pixel e dell'ultima in 100x100 pixel. Cambia inoltre la dimensione delle immagini dispari in 150x150 pixel.



Pseudo Elementi

Gli pseudo elementi in CSS sono delle parole chiave che si aggiungono ai selettori per modificare una parte specifica degli elementi HTML.

Per esempio, puoi usare gli pseudo elementi per aggiungere del contenuto prima o dopo il contenuto di un elemento, per cambiare lo stile della prima lettera o della prima riga di un elemento, per creare delle forme geometriche con il bordo o lo sfondo di un elemento, per evidenziare il testo selezionato da un utente, ecc...

La sintassi degli pseudo elementi è la seguente:

```
selettore::pseudo-elemento {  
    proprietà: valore;  
}
```

Il nome dello pseudo elemento è preceduto da due punti (:) e segue senza spazi il nome del selettore. Puoi usare gli pseudo elementi con tutti i selettori, come elementi, classi, id o attributi.

Pseudo elementi per contenuto generato

I pseudo elementi `::before` e `::after` ti permettono di inserire del contenuto prima o dopo il contenuto di un elemento. Il contenuto inserito è di tipo inline e non fa parte del documento HTML, ma solo del foglio di stile CSS. Per specificare il contenuto da inserire, devi usare la proprietà `content`.

Ecco alcuni esempi di come usare questi pseudo elementi per generare del contenuto:

Inserire un'immagine prima del contenuto di ogni elemento `<h1>`.

```
● ● ●  
h1::before {  
    content: url("logo.png");  
}
```



Inserire il testo “Nota:” dopo il contenuto di ogni elemento con classe .nota

```
● ● ●  
.nota::after {  
    content: "Nota:";  
}
```

Inserire delle virgolette aperte e chiuse intorno al contenuto di ogni elemento <q>

```
● ● ●  
q::before {  
    content: open-quote;  
}  
  
q::after {  
    content: close-quote;  
}
```

Gli pseudo elementi per il contenuto generato possono essere usati con qualsiasi tipo di elemento, ma devono avere sempre la proprietà content definita, altrimenti non avranno alcun effetto.

Pseudo elementi per la prima lettera e la prima riga

I pseudo elementi `::first-letter` e `::first-line` ti permettono di cambiare lo stile della prima lettera o della prima riga del testo contenuto in un elemento. Questi pseudo elementi sono utili per creare degli effetti tipografici, come le lettere capitali o le righe iniziali in grassetto. Ecco alcuni esempi di come usare questi pseudo elementi per personalizzare il test.



Cambiare colore e la dimensione della prima lettera di ogni elemento <p>

```
● ● ●  
p::first-letter {  
    color: red;  
    font-size: 2em;  
}
```

Cambiare colore e il tipo di carattere della prima riga di ogni elemento <p>

```
● ● ●  
p::first-line {  
    color: blue;  
    font-family: Arial, sans-serif;  
}
```

Gli pseudo elementi per la prima lettera e la prima riga possono essere usati solo con gli elementi di tipo blocco, come <p>, <div> o <h1>.

Pseudo elementi per forme geometriche

I pseudo elementi ::before e ::after possono essere usati anche per creare delle forme geometriche ad esempio con il bordo o con lo sfondo di un elemento. Questo metodo si basa sull'uso di alcune proprietà CSS, come width e height, border-radius, background, transform, ecc...

Ecco alcuni esempi di come usare queste proprietà per creare delle forme geometriche:



Crea un cerchio rosso con il bordo nero prima del contenuto di ogni elemento con classe .box

```
● ● ●  
.box::before {  
    content: "";  
    display: inline-block;  
    width: 50px;  
    height: 50px;  
    border: 2px solid black;  
    border-radius: 50%;  
    background: red;  
}
```

Crea un quadrato blu con il bordo giallo dopo il contenuto di ogni elemento .box

```
● ● ●  
.box::after {  
    content: "";  
    display: inline-block;  
    width: 50px;  
    height: 50px;  
    border: 2px solid yellow;  
    background: blue;  
}
```

Crea un triangolo verde con il bordo rosso prima del contenuto di ogni elemento .box

```
● ● ●  
.box::before {  
    content: "";  
    display: inline-block;  
    width: 0;  
    height: 0;  
    border-left: 25px solid transparent;  
    border-right: 25px solid transparent;  
    border-bottom: 50px solid green;  
}
```

Gli pseudo elementi per le forme geometriche devono avere sempre la proprietà `content` definita.

Pseudo elementi per il testo selezionato

Lo pseudo elemento `::selection` ti permette di cambiare lo stile del testo selezionato dall'utente con il mouse o con la tastiera. Questo pseudo elemento è utile per evidenziare il testo in modo diverso dal colore di default (solitamente blu). Ecco un esempio di come usare questo pseudo elemento per personalizzare il testo selezionato:

Cambiare il colore e lo sfondo del testo selezionato

```
● ● ●  
::selection {  
    color: white;  
    background: black;  
}
```



Lo pseudo elemento `::selection` è supportato da tutti i principali browser, ma con alcune differenze. Alcuni browser richiedono un prefisso, come `-moz-` per Firefox o `-webkit-` per Chrome e Safari. Inoltre, le proprietà CSS che si possono usare con questo pseudo elemento sono limitate a `color`, `background`, `cursor` e `outline`.



Esercizi

Crea una pagina HTML con un titolo, una paragrafo e un link. Cambia il colore e la dimensione della prima lettera del titolo, del paragrafo e del link con dei valori scelti da te.

Crea una pagina HTML con una lista non ordinata contenente cinque elementi di lista. Inserisci per gli elementi dispari un testo “dispari” prima del contenuto e per gli elementi pari un testo “pari”.

Crea una pagina HTML con un paragrafo e tre elementi `span` all'interno. Inserisci un testo dopo il contenuto `span`, usando il testo che preferisci.

Crea una pagina HTML con un `div` che contenga un testo di esempio. Cambia il colore di sfondo del testo quando viene selezionato dall'utente, usando i colori e gli sfondi che preferisci.

Crea una pagina HTML con un `div` che contiene un'immagine. Usa gli pseudo elementi per inserire un elemento nero semi-trasparente sopra l'immagine.



!important

La proprietà `!important` è un modo per dare più importanza a una regola CSS rispetto alle altre, e per sovrascrivere gli stili precedenti. Si usa aggiungendo `!important` dopo il valore della proprietà CSS, come in questo esempio:

```
● ● ●  
p {  
    color: blue;  
}  
  
p {  
    color: red !important;  
}
```

In questo caso, il paragrafo avrà il colore rosso, perché la seconda regola `!important` prevale sulla prima. `!important` si usa quando si vuole forzare l'applicazione di uno stile, ignorando le specificità e la cascata del CSS. Tuttavia, `!important` ha anche degli svantaggi, perché può creare dei problemi di consistenza e di manutenzione, e rendere il CSS più difficile da modificare e sovrascrivere. Per questo, è fondamentale usare `!important` solo come ultima risorsa, e di cercare invece di aumentare la specificità dei selettori e di riorganizzare il CSS per evitare i conflitti.



Transform

La proprietà `transform` in CSS permette di applicare delle trasformazioni 2D o 3D agli elementi di una pagina web, come rotazioni, scalatore, traslazioni o inclinazioni. Questa proprietà, unita ad altre proprietà, è utile per creare effetti visivi dinamici e animazioni.

Per usare la proprietà `transform`, devi specificare uno o più valori, che indicano il tipo e l'entità della trasformazione da applicare. Alcune delle funzioni più comuni sono:

`rotate (angolo)`: ruota l'elemento di un certo angolo attorno al suo centro.

`scale (x, y)`: scala l'elemento secondo i fattori x e y.

`translate (x, y)`: trasla l'elemento di una certa distanza lungo gli assi x e y.

`skew (x-angle, y-angle)`: inclina l'elemento secondo gli angoli x-angle e y-angle.

Puoi anche usare le varianti 3D di queste funzioni, aggiungendo il suffisso `3d` e specificando un valore per l'asse z.

Puoi combinare più funzioni di trasformazione separandole con uno spazio, in modo da ottenere una trasformazione composta.

L'ordine delle funzioni influisce sul risultato finale, in quanto le funzioni vengono moltiplicate da sinistra a destra.

La forma compatta della proprietà è:

`Transform: none | transform-function`

Dove `none` indica che non si vuole applicare nessuna trasformazione, e `transform-function` indica uno o più funzioni di trasformazione separate da spazi.



Ecco alcuni esempi che usano la proprietà `transform`:

Ruota un elemento di 45 gradi.

```
● ● ●  
.elemento {  
    transform: rotate(45deg);  
}
```

Scala un elemento del 50% in orizzontale e del 200% in verticale.

```
● ● ●  
.elemento {  
    transform: scale(0.5, 2);  
}
```

Trasla un elemento di 100px in orizzontale e di 50px in verticale.

```
● ● ●  
.elemento {  
    transform: translate(100px, 50px);  
}
```

Inclina un elemento di 30 gradi lungo l'asse x e di 15 gradi lungo l'asse y

```
● ● ●  
.elemento {  
    transform: skew(30deg, 15deg);  
}
```



Combina più funzioni di trasformazione.



```
.elemento {  
    transform: rotate(10deg) scale(1.5) translate(50px, 25px);  
}
```



Esercizi

Crea una pagina HTML con un elemento `div` che abbia un testo all'interno. Applica una trasformazione CSS al `div` in modo che sia ruotato di 45 gradi.

Crea una pagina HTML con un elemento `div` che abbia un'immagine all'interno. Applica una trasformazione CSS al `div` in modo che l'immagine sia scalata al doppio delle sue dimensioni.

Crea una pagina HTML con un elemento `div` di dimensioni 200x200 pixel e di colore rosso. Applica una trasformazione CSS al `div` in modo che sia traslato di 50 pixel in orizzontale e 30 pixel in verticale.

Crea una pagina HTML con un elemento `div` di dimensioni 200x100 pixel e di colore verde. Applica una trasformazione CSS al `div` in modo che sia inclinato di 20 gradi verso destra.



Object-fit

`object-fit` è una proprietà CSS che permette di definire come il contenuto di un elemento sostitutivo, come un'immagine o un video, dovrebbe essere ridimensionato per adattarsi al suo contenitore. Puoi anche modificare l'allineamento del contenuto dell'elemento sostitutivo usando la proprietà `object-position`.

`object-fit` funziona in questo modo, puoi scegliere tra diversi valori, che determinano come il contenuto si adatta al contenitore.

Alcuni valori possibili sono:

- `fill` il contenuto riempie tutto il contenitore, anche se per farlo deve essere stirato o schiacciato.
- `contain` il contenuto mantiene le sue proporzioni originali, ma si ridimensiona per entrare nel contenitore, anche se per farlo deve lasciare degli spazi vuoti.
- `cover` il contenuto mantiene le sue proporzioni originali, ma si ridimensiona per entrare nel contenitore, anche se per farlo deve essere tagliato.
- `none` il contenuto non si ridimensiona, ma rimane con le sue dimensioni originali, anche se per farlo deve uscire dal contenitore.

Vediamo un paio di esempi per capire meglio.

Immagina di avere un'immagine quadrata di 300x300 pixel e un contenitore rettangolare di 200x100 pixel. Se usi `object-fit: fill`; l'immagine si adatterà al contenitore, ma si deformerà, diventano un rettangolo. Se usi `object-fit: contain`; l'immagine si adatterà al contenitore, ma manterrà le sue proporzioni, lasciando degli spazi vuoti sopra e sotto. Se usi `object-fit: cover`; l'immagine si adatterà al contenitore, ma manterrà le sue proporzioni, tagliando le parti laterali. Se usi `object-fit: none`; l'immagine non si adatterà al contenitore, ma rimarrà con le sue dimensioni originali uscendo dal contenitore.



`object-fit` è una proprietà utile perché permette di gestire il ridimensionamento dei contenuti sostitutivi in modo flessibile e personalizzato, senza dover usare altri strumenti come il tag `<picture>` o le media query. Puoi usare `object-fit` per creare effetti visivi interessanti, come le immagini di sfondo, le miniature, le gallerie, ecc...



Transition

La proprietà CSS `transition` consente di creare effetti di transizione sugli elementi utilizzando solo i CSS.

Normalmente, quando il valore di una proprietà CSS cambia, il risultato viene renderizzato istantaneamente e l'elemento a cui è assegnata tale proprietà viene aggiornato immediatamente. Con le transizioni, invece, è possibile assegnare una durata (espressa in secondi) per il passaggio dal vecchio valore al nuovo valore.

Questo significa che possiamo creare effetti di transizione come il cambiamento del colore di sfondo di un contenitore al passaggio del mouse, o cambiare colore ad un testo, o modificare la larghezza, altezza, posizione e in tante altre occasioni.

La proprietà `transition` accetta quattro valori:

- `transition-property` specifica la proprietà CSS su cui applicare l'effetto di transizione. Ad esempio, `background-color`, `color`, `width`, `height`, ecc..
- `transition-duration` specifica la durata dell'effetto di transizione, espressa in secondi o millisecondi. Ad esempio `2s`, `500ms`, ecc...
- `transition-timing-function` specifica la funzione di temporizzazione dell'effetto di transizione. Ad esempio `linear`, `ease`, `ease-in`, `ease-out`, `ease-in-out`, ecc...
- `transition-delay` specifica il ritardo prima che l'effetto di transizione inizi, espressa in secondi o millisecondi. Ad esempio, `2s`, `500ms`, ecc...



Vediamo alcuni esempi.

Creare un effetto di transizione sul colore di sfondo di un contenitore al passaggio del mouse.

```
● ● ●  
.container {  
    background-color: #98d925;  
    transition-property: background-color;  
    transition-duration: 2s;  
}  
  
.container:hover {  
    background-color: #ff5c00;  
}
```

Creare un effetto di transizione sulla larghezza di un contenitore al passaggio del mouse.

```
● ● ●  
.container {  
    width: 200px;  
    transition-property: width;  
    transition-duration: 2s;  
}  
  
.container:hover {  
    width: 400px;  
}
```

La proprietà `transition` può essere scritta in forma ridotta utilizzando la seguente sintassi:

```
transition: [property] [duration] [time-function]  
[delay];
```



dove `[property]` è la proprietà CSS su cui si vuole applicare la transizione, `[duration]` è la durata della transizione in secondi o millisecondi, `[time-function]` è la funzione di temporizzazione dell'effetto di transizione e `[delay]` è il ritardo prima che l'effetto di transizione inizi.

Ecco un esempio di come utilizzare la proprietà `transition` in forma ridotta



```
.container {  
    background-color: #98d295;  
    transition: background-color 0.5s ease-in-out 1s;  
}  
  
.container:hover {  
    background-color: #ff5c00;  
}
```

La funzione di temporizzazione della proprietà `transition` consente di definire l'accelerazione dell'effetto di transizione.

Alcuni valori sono:

- `linear` questa funzione di temporizzazione produce un'accelerazione costante durante l'intera durata dell'effetto di transizione.
- `ease-in` produce un'accelerazione graduale all'inizio dell'effetto di transizione.
- `ease-out` produce un'accelerazione graduale alla fine dell'effetto di transizione.
- `ease-in-out` produce un'accelerazione graduale all'inizio e alla fine dell'effetto di transizione.
- `cubic-bezier` consente di definire una curva di accelerazione personalizzata utilizzando quattro valori numerici.



Animation

La proprietà CSS `animation` consente di creare animazioni sugli oggetti utilizzando solo i CSS. Questo significa che possiamo creare animazioni come il movimento di un'immagine, il cambio di colore di sfondo, la larghezza, la posizione, la trasparenza, ecc...

Normalmente, quando si utilizza la proprietà `animation`, si definisce una sequenza di animazione chiamata `@keyframes`. La sequenza di animazioni inizia con il valore `from` e termina con il valore `to`. Tuttavia, è possibile specificare qualsiasi numero di keyframe utilizzando la sintassi percentuale. Ad esempio, 50% indica il punto a metà strada lungo la sequenza di animazione.

La proprietà `animation` accetta otto valori:

- `animation-name` specifica il nome dell'animazione definita con il `@keyframes`.
- `animation-timing-function` specifica la funzione di temporizzazione dell'animazione.
- `animation-delay` specifica il ritardo prima che l'animazione inizi, espressa in secondi o millisecondi.
- `animation-iteration-count` specifica il numero di volte che l'animazione può essere ripetuta.
- `animation-direction` specifica la direzione dell'animazione.
- `animation-fill-mode` specifica lo stato finale dell'animazione.
- `animation-play-state` specifica se l'animazione è in esecuzione o in pausa.

La proprietà `animation` può essere scritta in forma ridotta utilizzando la seguente sintassi:

```
animation: [name] [duration] [timing-function] [delay]  
[iteration-count] [direction] [fill-mode] [play-state];
```



Ecco alcuni esempi di come utilizzare la proprietà `animation`.

Creare un'animazione di rotazione di un'immagine.

```
● ● ● ● ●  
img {  
    animation: rotate 2s infinite linear;  
}  
  
@keyframes rotate {  
    from {  
        transform: rotate(0deg);  
    }  
    to {  
        transform: rotate(360deg);  
    }  
}
```

Creare un'animazione di scorrimento su un'immagine.

```
● ● ● ● ●  
img {  
    animation: slide 2s infinite linear;  
}  
  
@keyframes slide {  
    from {  
        left: 0px;  
    }  
    to {  
        left: 100px;  
    }  
}
```



Esercizi

Crea una pagina HTML con un elemento `div` che abbia un colore di sfondo rosso. Applica una transizione CSS al colore di sfondo in modo che cambia gradualmente in blu quando il mouse vi passa sopra.

Crea una pagina HTML con un elemento `div` che abbia un'immagine all'interno. Applica una transizione CSS alla dimensione dell'immagine in modo che si riduce gradualmente a metà quando il mouse vi passa sopra.

Crea una pagina HTML con un elemento `div` di colore verde. Applica una transizione CSS alla posizione del `div` in modo che si sposta gradualmente di 100 pixel verso destra quando il mouse vi passa sopra.

Crea una pagina HTML con un elemento `div` di colore giallo. Applica una animazione CSS al `div` in modo che ruota continuamente di 360 gradi.

Crea una pagina HTML con un elemento `div` di colore arancione. Applica una animazione CSS al `div` in modo che cambia il colore di sfondo in modo ciclico tra rosso, verde e blu.



Media query

Le media query sono funzionalità di CSS che consentono di applicare stili CSS in base alle caratteristiche del dispositivo o del browser utilizzato per visualizzare la pagina. Le media query possono essere utilizzate per controllare molte cose, come la larghezza e l'altezza della viewport, l'orientamento, la risoluzione dello schermo, ecc... Le media query sono utilizzate per creare siti web responsivi, che si adattano automaticamente alle dimensioni dello schermo del dispositivo utilizzato per visualizzare la pagina.

Il responsive design è un approccio di progettazione web che consente di creare siti web che si adattano automaticamente alle dimensioni dello schermo del dispositivo utilizzato per visualizzare la pagina.

In pratica, questo significa che il layout del sito web viene modificato in modo dinamico in base alle dimensioni dello schermo del dispositivo, in modo da garantire una visualizzazione ottimale dei contenuti e un'esperienza utente unica.

Il concetto di “mobile first” invece è una filosofia di progettazione web che prevede di progettare prima per i dispositivi mobile e poi per desktop. Questo approccio si basa sul fatto che sempre più persone utilizzano dispositivi mobile e per navigare e che i siti web devono essere progettati per fornire un'esperienza utente ottimale su questi dispositivi.

Le media query sono uno strumento fondamentale per la creazione di siti web responsive e per il concetto mobile first. Infatti, consentono di applicare stili CSS in base alle caratteristiche del dispositivo.

La sintassi delle media query è la seguente:

```
● ● ●  
 @media not|only mediatype and (media feature) {  
   /* CSS rules */  
 }
```



Dove `mediatype` è il tipo di media, ad esempio `screen`, `print`, ecc...
`media feature` è una caratteristica del dispositivo o del browser, ad esempio `width`, `height`, `orientation`, `resolution`, ecc... `not` e `only` sono parole chiave che specificano se la media query deve essere applicata solo al tipo di media specificato.

Ecco alcuni esempi:

Applicare stili solo ai dispositivi con una larghezza massima di 480px

```
● ● ●  
 @media only screen and (max-width: 480px) {  
     /* regole css */  
 }
```

Cambiare colore di sfondo di un `div .container` solo ai dispositivi con una larghezza minima di 768px

```
● ● ●  
 @media only screen and (min-width: 768px) {  
     .container {  
         background-color: black;  
     }  
 }
```



Esercizi

Crea una pagina HTML con un paragrafo. Applica una media query CSS per cambiare il colore del testo del paragrafo quando la larghezza della finestra del browser è inferiore a 600 pixel.

Crea una pagina HTML con un'immagine. Applica una media query CSS per ridurre le dimensioni dell'immagine a 50% della larghezza originale quando la larghezza della finestra del browser è inferiore a 800 pixel.

Crea una pagina HTML con una lista non ordinata consentente tre elementi di lista. Applica una media query CSS per nascondere il primo e l'ultimo elemento della lista quando la finestra del browser è inferiore a 600 pixel.

Crea una pagina HTML con un titolo. Applica una media query CSS per cambiare dimensione del testo del titolo a 48 pixel quando la larghezza della finestra del browser è maggiore di 700px;

Crea una pagina HTML con due elementi div, uno di dimensioni 300x300 pixel e di colore rosso e l'altro di dimensioni 200x100 pixel e di colore blue. Applica una media query CSS per non mostrare il secondo div quando la larghezza della finestra del browser è minore di 600 pixel.



Flexbox

flexbox è un modulo di CSS che ti permette di posizionare e allineare gli elementi in modo dinamico, adattandosi alle diverse dimensioni dello schermo e ai diversi flussi di testo. Con flexbox puoi creare layout complessi con poco codice.

flexbox funziona in una sola dimensione per volta, o in riga o in colonna ed è supportato dai browser già moderni, ma potrebbe richiedere dei prefissi per alcuni di essi.

Per usare flexbox, devi prima definire un contenitore flessibile (flex container) con la proprietà `display: flex;` o `display: inline-flex;`. Questo crea un contenitore flessibile per tutti i suoi figli diretti, che diventano degli elementi flessibili (flex items).

Il contenitore flessibile ha due assi: il main axis, che segue la direzione dei flex items, e il cross axis, che è perpendicolare al main axis. La direzione del main axis si può cambiare con la proprietà `flex-direction`, che accetta i seguenti valori:

```
● ● ●  
.flex-container {  
  display: flex;  
  flex-direction: row;  
}
```

`flex-direction: row` (default)

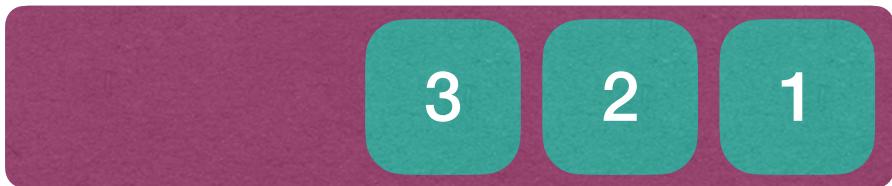
da sinistra a destra.

1 2 3



flex-direction: row-reverse

da destra a sinistra.



flex-direction: column

dall'alto verso il basso.



flex-direction: column-reverse

dal basso verso l'alto.



La proprietà **flex-wrap**, invece, controlla se i flex items devono stare su una sola linea o se possono andare a capo, occupando più linee.

I valori possibili sono:

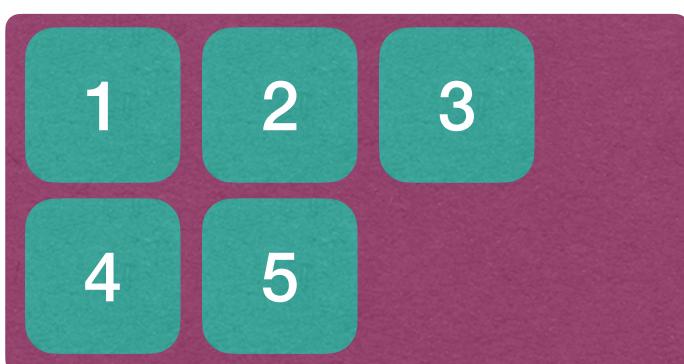
flex-wrap: nowrap (default)

tutti i flex items stanno su una sola linea.



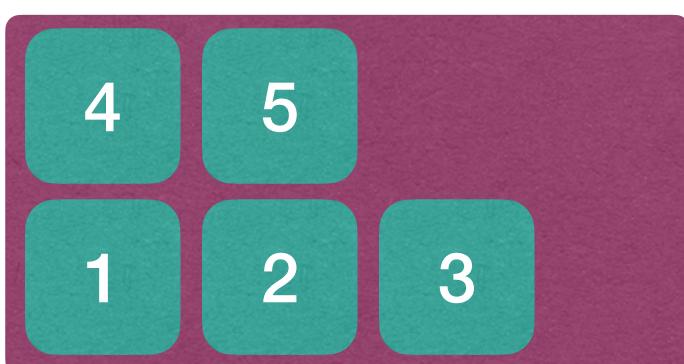
flex-wrap: wrap

i flex items vanno a capo da sopra verso sotto.



flex-wrap: wrap-reverse

i flex items vanno a capo da sotto a sopra.



La proprietà **flex-flow** è una scorciatoia per `flex-direction` e `flex-wrap`, e accetta due valori separati da uno spazio. Il valore di default è `row nowrap`.



```
.flex-container {  
    display: flex;  
    flex-flow: row wrap;  
}
```

La proprietà **justify-content** definisce l'allineamento dei flex items lungo il main axis, e aiuta a distribuire lo spazio libero rimanente quando i flex items sono inflessibili o hanno raggiunto la loro dimensione massima. Può anche influenzare l'allineamento dei flex items quando sforano la linea. Alcuni valori possibili sono:

justify-content: flex-start;

i flex items sono spinti verso l'inizio del main axis.



justify-content: flex-end;

i flex items sono spinti verso la fine del main axis.



justify-content: center;

i flex items sono centrati lungo il main axis.



justify-content: space-between;

i flex items sono distribuiti uniformemente lungo il main axis, con il primo elemento all'inizio e l'ultimo elemento alla fine.



justify-content: space-around;

i flex items sono distribuiti uniformemente lungo il main axis, con lo stesso spazio tra loro e i lati.

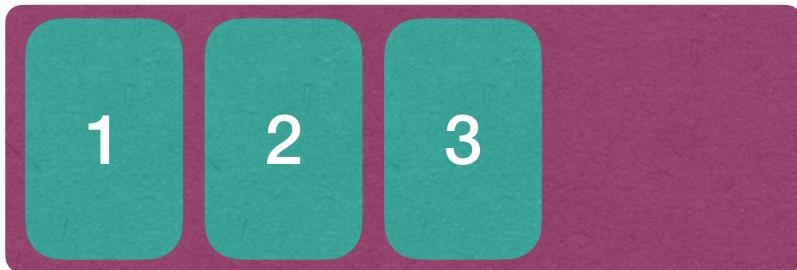


La proprietà **align-items** definisce l'allineamento dei flex items lungo il cross axis, e aiuta a distribuire lo spazio libero rimanente quando i flex items hanno altezze diverse.

Alcuni valori possibili sono:

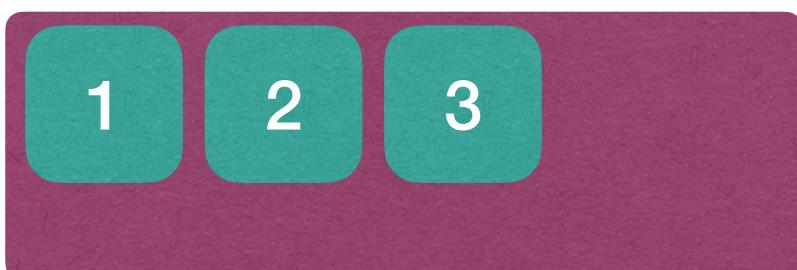
align-items: stretch;

i flex items si estendono per occupare tutto lo spazio disponibile lungo il cross axis.



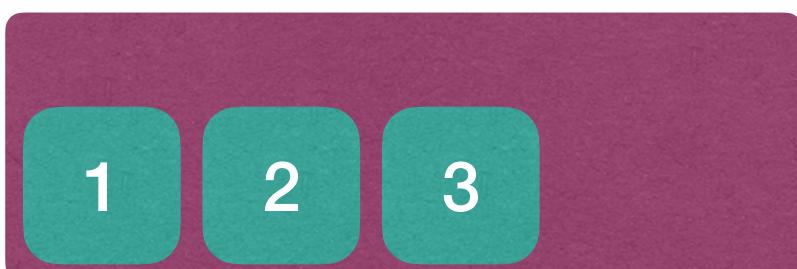
align-items: flex-start;

i flex items sono spinti verso l'inizio del cross axis.

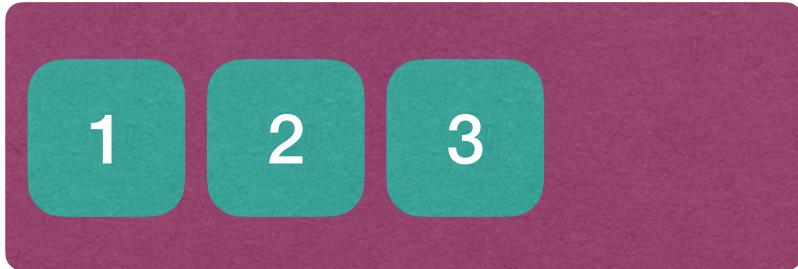


align-items: flex-end;

i flex items sono spinti verso la fine del cross axis.



align-items: center - i flex items sono centrati lungo il cross axis.



La proprietà **align-content**, invece, definisce l'allineamento delle linee di flex items lungo il cross axis, e aiuta a distribuire lo spazio libero rimanente quando il contenitore flessibile è più grande della somma delle linee. Questa proprietà ha effetto solo se ci sono più linee di flex items, altrimenti è ignorata. I valori possibili sono gli stessi di `justify-content`.

Esercizi

Crea una pagina HTML con tre elementi `div`. Applica un flex container ai tre elementi in modo che siano disposti in riga.

Crea una pagina HTML con quattro elementi `div`. Applica un flex container ai quattro elementi in modo che siano disposti in colonna e si adattino alla larghezza del contenitore.

Crea una pagina HTML con cinque elementi `div`. Applica un flex container ai cinque elementi in modo che siano disposti in riga e allineati al centro verticalmente.

Crea una pagina HTML con tre elementi `div`. Applica un flex container ai tre elementi in modo che siano disposti in colonna e allineati al centro orizzontalmente.

Crea una pagina HTML con quattro elementi `div`. Applica un flex container ai quattro elementi in modo che siano disposti in riga e separati da uno spazio uguale tra loro.



Organizzare i file CSS

Dividere il CSS in diversi file può avere diversi vantaggi come migliorare l'organizzazione e la manutenibilità del codice, separando gli stili per diversi componenti, pagine e funzionalità e aumentare la modularità e la riusabilità del codice, evitando la duplicazione e la sovrascrittura degli stili.

Per dividere il CSS in diversi file, puoi usare la regola `@import`, che ti permette di importare un file CSS in un altro file CSS.

Ad esempio:

```
/* File style.css */
@import url("reset.css");
@import url("layout.css");
@import url("typography.css");
```

In questo modo, il file `style.css` include gli stili definiti nei file `reset.css`, `layout.css` e `typography.css`. Puoi usare qualsiasi nome per i file, purché siano validi e coerenti.



Alcuni consigli

Per usare il CSS in modo efficace e organizzato, ci sono alcune best practices che dovresti seguire.

Organizza il tuo foglio di stile. Usa una struttura coerente e ordinata per il tuo CSS, con interruzioni di riga, commenti e sezioni che ti aiutino a capire il codice a colpo d'occhio.

Evita il CSS in linea. Inserire stile direttamente negli elementi HTML rende il codice più difficile da modificare e riutilizzare. Usa invece dei fogli di stile esterni, collegati alle pagine HTML con il tag `<link>`. Questo ti permette di separare la struttura dello stile e di applicare lo stesso stile a più pagine. Riduci al minimo il foglio di stile. Un foglio di stile troppo lungo e complesso può rallentare il caricamento delle pagine e creare confusione.

Inizia con un reset. I browser hanno dei valori di default per gli elementi HTML, che possono variare da uno all'altro. Per evitare delle differenze di stile indesiderate, usa un reset CSS per annullare i valori di default e avere una base uniforme su cui lavorare.

Usa le classi e gli ID in modo appropriato. Come abbiamo visto le classi e gli ID sono degli attributi che puoi assegnare agli elementi HTML, per selezionarli con il CSS. Le classi possono essere usate per più elementi, mentre gli ID devono essere unici. Usa le classi per gli stili generali e gli ID per gli stili specifici. Segui delle convenzioni di denominazione chiare e descrittive, come BEM o SMACSS, per rendere il tuo CSS più leggibile e scalabile.

Evita la ridondanza. Non ripetere lo stesso stile per più elementi, ma usa la regola della cascata del CSS per ereditare gli stili dai genitori ai figli. Usa i selettori di gruppo per applicare uno stile a più elementi separati da una virgola. Usa le proprietà separate da una virgola. Usa le proprietà abbreviate per scrivere meno codice.

Importa correttamente i font. Se vuoi usare dei font diversi da quelli default, devi importarli nel tuo CSS con la regola `@import` o il tag `<link>`. Puoi usare dei font gratuiti da Google Fonts. Assicurati di specificare il formato del



font, il peso e lo stile, e di usare la proprietà `font-family` per applicare il font agli elementi desiderati. Usa anche la proprietà `font-display` per controllare come il font viene caricato e visualizzato.

Rendi il CSS accessibile. Il CSS non solo influisce sull'aspetto delle pagine, ma anche sull'esperienza utente e sull'accessibilità. Usa dei colori contrastanti e leggibili per il testo e lo sfondo, e evita di usare il colore come unico mezzo per trasmettere informazioni.

Usa delle unità relative come `em`, `rem` o `%` invece di usare unità assolute come `px` per adattare il tuo CSS alle dimensioni dello schermo e alle preferenze dell'utente.

Usa le media query per creare dei layout responsive che si adattano a diverse risoluzioni e dispositivi.

Evita il tag `!important`. Il tag `!important` è un modo per forzare l'applicazione di uno stile, ignorando la specificità e la cascata del CSS. Tuttavia, questo può creare dei problemi di consistenza e di manutenzione, e rendere il tuo CSS più difficile da sovrascrivere. Usa il tag `!important` solo come ultima risorsa, e cerca invece di aumentare la specificità dei tuoi selettori o di riorganizzare il tuo CSS per evitare conflitti.

Per quanto riguarda invece la compatibilità dei browser in CSS, che specifica che il tuo codice CSS funziona correttamente e in modo uniforme su diversi browser web, come Chrome, Firefox, Safari, Opera, Edge, ecc... è importante verificarla perché i browser hanno delle differenze nell'interpretare e applicare le regole CSS, soprattutto per le proprietà più recenti o sperimentali. Per garantire la compatibilità dei browser in CSS, ci sono alcune best practices che puoi seguire.

Usa i prefissi dei vendor. I prefissi dei vendor sono dei prefissi che i browser aggiungono alle proprietà CSS non standardizzate o in fase di sviluppo, per indicare che sono specifiche per quel browser. Ad esempio il prefisso `-webkit-` si usa per Chrome e Safari, il prefisso `-moz-` per Firefox, il prefisso `-ms-` per Edge. Usando i prefissi dei vendor, puoi assicurarti che le proprietà CSS sono riconosciute e applicate da tutti i browser che le supportano.

