CSS

Scritto da Febogamedev

CSS, acronimo di Cascading Style Sheets, in italiano fogli di stile a cascata, è il linguaggio utilizzato per descrivere la presentazione di documenti strutturati, come le pagine *HTML*. (W3C CSS Working Group, 2026b)
Un file che contiene codice CSS utilizza l'estensione .css.

CSS permette di controllare, tra gli altri aspetti:
- Colori
- Tipografia
- Dimensioni
- Spaziature
- Bordi e sfondi
- Disposizione degli elementi
- Adattamento ai diversi schermi
- Stati interattivi
- Transizioni e animazioni
- Presentazione su schermo o stampa

HTML descrive principalmente la struttura e il significato dei contenuti, mentre *CSS* stabilisce come questi contenuti devono essere rappresentati. (W3C CSS Working Group, 2026b)
Possiamo sintetizzare la relazione in questo modo:
- *HTML* → che cosa rappresenta il contenuto.
- *CSS* → come viene presentato il contenuto.

Questa separazione permette di modificare l'aspetto della pagina senza riscriverne la struttura.

Per esempio, lo stesso documento *HTML* può essere presentato:
- Con colori e caratteri differenti.
- In una o più colonne.
- Su uno smartphone o su uno schermo ampio.
- In modalità chiara o scura.
- Sullo schermo oppure in stampa.

Un foglio di stile è un insieme di regole *CSS*. (W3C CSS Working Group, 2021b)
Ogni regola stabilisce:
- Quali elementi devono essere selezionati.
- Quali proprietà devono essere applicate.
- Quali valori devono essere assegnati a queste proprietà.

Per esempio:

```css
a {
  color: darkblue;
  font-weight: 700;
  text-decoration-thickness: 2px;
}
```

Questa regola seleziona tutti gli elementi *HTML* <a> e assegna al testo il colore darkblue, un peso del carattere pari a 700 e uno spessore di 2px alle eventuali linee di decorazione, come sottolineature, sopralineature e barrature.

Come abbiamo visto dalla regola, il CSS non modifica il significato dell'elemento (per esempio un collegamento rimane un collegamento), ma ne modifica la presentazione.

Un foglio di stile esterno può essere associato al documento *HTML* attraverso l'elemento <link> normalmente inserito all'interno di <head>. (WHATWG, n.d.)

```html
<head>
  <link rel="stylesheet" href="style.css">
</head>
```

- L'attributo "rel" con valore "stylesheet" indica che la risorsa collegata è un foglio di stile. (WHATWG, n.d.)
- L'attributo "href" con valore "style.css" indica l'*URL* del file *CSS* (in questo esempio l'*URL* è relativo e viene risolto rispetto all'indirizzo di base del documento *HTML*). (WHATWG, n.d.)

Utilizzare un file esterno per "decorare" il file *HTML* permette di applicare lo stesso sistema visivo a molte pagine.
Una modifica al foglio di stile può quindi aggiornare simultaneamente l'aspetto di un intero sito.
CSS può essere inserito anche direttamente nel documento *HTML*, ma i fogli esterni facilitano generalmente organizzazione, riutilizzo e manutenzione. (W3C CSS Working Group, 2026b)

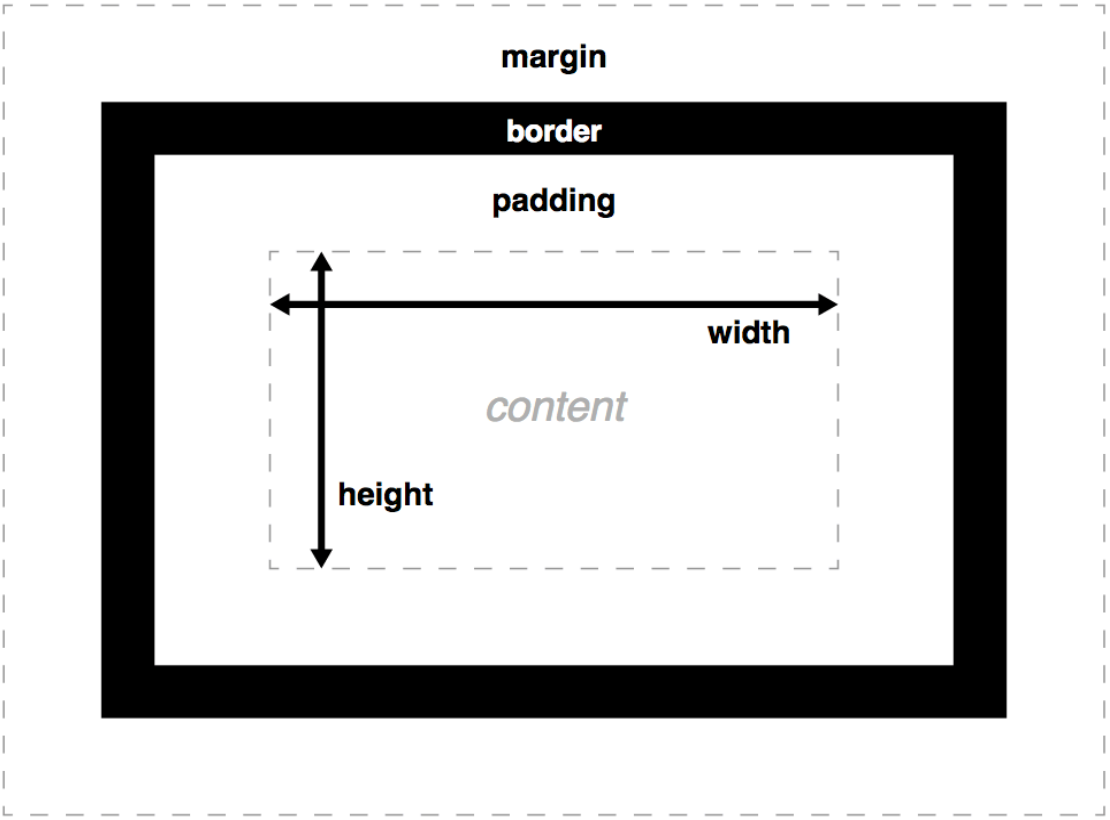
Box model

Per rappresentare visivamente un documento, il browser genera dei riquadri, chiamati box, associati agli elementi *HTML* che partecipano alla disposizione della pagina. (W3C CSS Working Group, 2026a)

Il modello fondamentale utilizzato per descrivere questi riquadri prende il nome di **box model**. (W3C CSS Working Group, 2024b)
Ogni box è formato da quattro aree annidate una intorno all'altra. Procedendo dall'esterno verso l'interno troviamo: (W3C CSS Working Group, 2024b)

- Margine
- Bordo
- Padding

- Contenuto



Contenuto
La content area è la parte più interna del box. Al suo interno vengono disposti il testo, le immagini, i video e i box generati dagli eventuali elementi discendenti. Non comprende padding, bordo e margine.

La sua larghezza e la sua altezza non sono sempre indicate direttamente, ma vengono calcolate dal browser considerando diversi fattori.

- Il contenuto: quando le dimensioni sono automatiche, il contenuto può contribuire a determinarle. Per esempio, l'altezza di un paragrafo aumenta quando il testo occupa più righe, mentre un'immagine può partire dalle proprie dimensioni naturali. (W3C CSS Working Group, 2021a)
- Le proprietà width e height: permettono di indicare rispettivamente la larghezza e l'altezza dell'elemento. Con il valore predefinito box-sizing: content-box, una dichiarazione come width: 300px assegna alla content area una larghezza di 300px. Padding e bordo vengono aggiunti all'esterno di questa misura. (W3C CSS Working Group, 2021a)
- Il containing block: quando una dimensione è espressa in percentuale, il browser deve calcolarla rispetto a un rettangolo di riferimento chiamato containing block. Nel flusso normale, questo rettangolo coincide spesso con la content area dell'elemento genitore. Se il containing block è largo 800px, un elemento figlio con width: 50% avrà una larghezza di 400px. (W3C CSS Working Group, 2021a)
- Il sistema di layout: il modo in cui gli elementi sono organizzati può influenzarne le dimensioni.

    - Nel flusso normale, gli elementi block con larghezza automatica occupano generalmente lo spazio orizzontale disponibile.
    - Con Flexbox, gli elementi figli possono allargarsi o restringersi per condividere lo spazio lungo una riga o una colonna. (W3C CSS Working Group, 2025a)
    - Con Grid vengono invece inseriti in una struttura di righe e colonne, le cui dimensioni influenzano quelle degli elementi contenuti. (W3C CSS Working Group, 2025b)
- I limiti minimi e massimi: le proprietà min-width, max-width, min-height e max-height impediscono all'elemento di diventare più piccolo o più grande dei limiti stabiliti. (W3C CSS Working Group, 2021a)

Questi fattori possono agire contemporaneamente. Per esempio, un elemento può avere width: 50% e max-width: 600px: occuperà metà della larghezza del containing block, senza però superare i 600 px.

È importante notare che la proprietà box-sizing stabilisce quali aree devono essere comprese nelle dimensioni assegnate attraverso width e height. (W3C CSS Working Group, 2021a)

- Con il valore predefinito content-box, width e height si riferiscono soltanto alla content area:
- Con border-box, invece, la misura comprende contenuto, padding e bordo:

Padding

Il padding è lo spazio interno compreso tra il contenuto e il bordo. Aumentarlo allontana il contenuto dal bordo dell'elemento. Lo sfondo viene normalmente disegnato anche nell'area del padding. (W3C CSS Working Group, 2024a)

È possibile controllare separatamente i quattro lati attraverso:

- padding-top
- padding-right
- padding-bottom
- padding-left

Bordo

Il bordo circonda il contenuto e il padding. Può essere controllato attraverso il suo spessore, stile e colore. La proprietà border-radius permette inoltre di arrotondarne gli angoli. (W3C CSS Working Group, 2024a)

Margine

Il margine è lo spazio esterno al bordo, che separa il box dagli elementi adiacenti. (W3C CSS Working Group, 2024b)

È trasparente: lo sfondo dell'elemento non lo copre. (W3C CSS Working Group, 2024b)

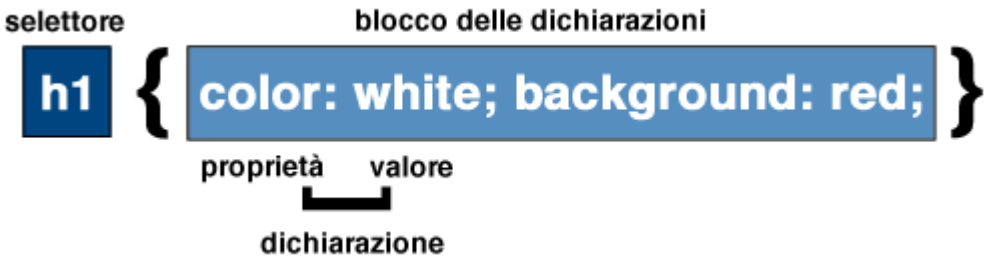
Come per il padding, è possibile controllare separatamente i suoi quattro lati mediante:

- margin-top
- margin-right
- margin-bottom
- margin-left

Regole

Una regola di stile *CSS* è composta da: (W3C CSS Working Group, 2021b)

- un selettore
- un blocco di dichiarazioni



I selettori indicano quali elementi *HTML* sono interessati dalla regola. (W3C CSS Working Group, 2026d)
I selettori possono riferirsi anche a classi, identificatori, attributi, relazioni tra elementi e stati dell'interfaccia.

Esempio

```
h1 {
  color: white;
  background-color: darkred;
}
```

Questa regola viene applicata a tutti gli elementi <h1>.

Il blocco delle dichiarazioni è delimitato rispetto al selettore e alle altre regole da due parentesi graffe, la prima di apertura e la seconda di chiusura. Al suo interno possono trovare posto più dichiarazioni. (W3C CSS Working Group, 2021b)

Una dichiarazione è composta da una coppia:

- proprietà
- valore

Ogni dichiarazione è composta da una **proprietà** e da un **valore**, separati dai due punti. (W3C CSS Working Group, 2021b)
La proprietà indica l'aspetto sul quale intervenire, mentre il valore specifica come deve essere impostato.

Esempio

```css
h1 {
  color: white;
  background-color: darkred;
}
```

Questa blocco delle dichiarazioni ha due dichiarazioni
- Nella prima dichiarazione color è la proprietà e white è il valore.
- Nella seconda dichiarazione background-color è la proprietà e darkred è il valore.

Le dichiarazioni vanno invece separate con un punto e virgola. (W3C CSS Working Group, 2021b)
Non è obbligatorio, ma è buona norma mettere il punto e virgola anche dopo l'ultima dichiarazione del blocco.

Alcune proprietà, chiamate **proprietà abbreviate** o shorthand, permettono di impostare contemporaneamente diverse proprietà correlate.

Esempio

```css
div {
  margin: 10px 5px 20px 0;
}
```

margin è una shorthand che imposta, nell'ordine, i margini superiore, destro, inferiore e sinistro.
La regola equivale a:

```css
div {
  margin-top: 10px;
  margin-right: 5px;
  margin-bottom: 20px;
  margin-left: 0;
}
```

I selettori di attributo individuano gli elementi HTML in base alla presenza o al valore di uno dei loro attributi. (W3C CSS Working Group, 2026d)
Per esempio, [disabled] seleziona tutti gli elementi che possiedono l'attributo disabled, mentre input[type="email"] seleziona soltanto gli elementi <input> il cui attributo type ha valore email.
È possibile anche selezionare valori che iniziano, terminano o contengono una determinata sequenza. Per esempio: (W3C CSS Working Group, 2026d)
- a[href^="https"] individua i collegamenti il cui indirizzo inizia con https.
- a[href$=".pdf"] quelli che terminano con .pdf e a[href*="wikipedia"] quelli che contengono la parola wikipedia.

Esempio

```
input[type="email"] {
    border: 2px solid blue;
}
```

Questa regola si applica agli elementi <input> il cui attributo type ha valore email.

Gli pseudo-elementi sono componenti che possono essere utilizzati all'interno di un selettore per rappresentare una parte o un contenuto associato all'elemento selezionato senza aggiungere nuovi elementi al codice *HTML*. (W3C CSS Working Group, 2025c)
Essi si scrivono generalmente con due punti "::".
Per esempio, "::first-letter" seleziona la prima lettera di un testo, "::first-line" la prima riga, mentre "::before" e "::after" permettono di generare contenuti rispettivamente prima e dopo il contenuto dell'elemento. (W3C CSS Working Group, 2025c)
Gli pseudo-elementi non rappresentano veri elementi presenti nel documento HTML, ma parti o contenuti creati durante la rappresentazione grafica della pagina.

Esempio

```
p::first-letter {
    color: red;
    font-size: 40px;
    font-weight: bold;
}
```

Questa regola seleziona e formatta soltanto la prima lettera dell'elemento p. Tale lettera verrà mostrata più grande, rossa e in grassetto.

Diversi selettori semplici possono essere combinati per individuare con maggiore precisione gli elementi. (W3C CSS Working Group, 2026d)
Quando vengono uniti senza un combinatore, come in input[type="email"], le condizioni si riferiscono allo stesso elemento.
Lo spazio è invece un **combinatore di discendenza** e richiede che l'elemento indicato a destra si trovi all'interno di quello indicato a sinistra. (W3C CSS Working Group, 2026d)
Uno pseudo-elemento viene normalmente collocato alla fine del selettore e individua una parte dell'elemento selezionato.

Esempio

```
form input[type="email"]::placeholder {
    color: blue;
    font-style: italic;
}
```

Questo selettore individua lo pseudo-elemento ::placeholder degli elementi <input> con attributo type="email" che si trovano all'interno di un elemento <form>.

Quindi ricapitoliamo:

- Il *CSS* è formato da regole.
- Ogni regola è formata da un selettore e da un blocco di dichiarazioni.
- Ogni dichiarazione è formata da una proprietà e da un valore.

Di seguito è riportato lo schema gerarchico:

```
Foglio di stile
└── Regola di stile 1
    ├── Selettore
    └── Blocco delle dichiarazioni
        └── Dichiarazione
            ├── Proprietà
            └── Valore
└── Regola di stile 2
    ├── ....
```

ID e classi

Per applicare gli stili a elementi specifici, CSS mette a disposizione diversi tipi di selettori. Tra i più utilizzati troviamo: (W3C CSS Working Group, 2026d)

- **I selettori di classe**, che individuano gli elementi in base all'attributo HTML class;
- **I selettori di ID**, che individuano un elemento in base all'attributo HTML id;
- **Le pseudo-classi**, che individuano gli elementi in base a una condizione, una posizione o uno stato, senza richiedere l'aggiunta di un attributo nel codice HTML.

I selettori di classe

L'attributo "class" permette di assegnare uno stesso nome a più elementi HTML. (WHATWG, n.d.)

Esempio:

```html
<article class="card">
  <h2>Primo progetto</h2>
</article>

<article class="card">
  <h2>Secondo progetto</h2>
</article>
```

Entrambi gli elementi appartengono alla classe card.
In CSS, una classe viene selezionata scrivendo il suo nome preceduto da un punto:

```css
.card {
  padding: 1rem;
  border: 1px solid gray;
  border-radius: 0.5rem;
}
```

La regola viene applicata a tutti gli elementi che possiedono la classe card.

Le classi sono quindi particolarmente adatte per definire componenti o stili riutilizzabili, come card, pulsanti, messaggi, intestazioni, griglie, varianti cromatiche, ecc.

Uno stesso elemento può possedere più classi, separate da uno spazio: (WHATWG, n.d.)

```html
<a class="pulsante pulsante-evidenziato" href="/contatti">
  Contatti
</a>
```

In questo caso l'elemento appartiene sia alla classe "pulsante" sia alla classe "pulsante-evidenziato".

Sotto si vede cosa fanno queste classi:

```css
.pulsante {
  display: inline-block;
  padding: 0.75rem 1rem;
  text-decoration: none;
}

.pulsante-evidenziato {
  background-color: darkblue;
  color: white;
}
```

- La prima classe definisce le caratteristiche comuni a tutti i pulsanti.
- La seconda aggiunge una particolare variante visiva.

Questo sistema permette di combinare regole differenti senza dover creare una classe specifica per ogni possibile combinazione.

I selettori di ID
L'attributo "id" assegna un identificatore a un elemento. (WHATWG, n.d.)

```html
<header id="intestazione-principale">
```

```
  ...
</header>
```

Il valore di un attributo “id” deve essere unico all’interno dello stesso documento *HTML*. (WHATWG, n.d.)
Non è quindi corretto assegnare lo stesso identificatore a due elementi.

In CSS, un “id” viene selezionato scrivendo il suo nome preceduto dal simbolo #:

```
#intestazione-principale {
  position: sticky;
  top: 0;
}
```

La regola viene applicata all’elemento il cui “id” è intestazione-principale.

Un “id” non viene utilizzato soltanto dal *CSS*.
Può servire anche come destinazione di un frammento *URL*. (WHATWG, n.d.)
Per esempio questo elemento ha l’attributo “id”:

```
<section id="contatti">
  <h2>Contatti</h2>
</section>
```

Il seguente collegamento, che è un link, conduce alla sezione il cui id ha valore contatti

```
<a href="#contatti">Vai ai contatti</a>
```

L’*URL* di questo link potrebbe apparire così: https://example.com/#contatti
La parte #contatti è il frammento *dell’URL* e corrisponde all’elemento che possiede l’id “contatti”. (WHATWG, n.d.)

Un id può anche mettere in relazione elementi HTML.
Per esempio, il valore dell’attributo for di un’etichetta deve corrispondere all’id del campo al quale l’etichetta si riferisce: (WHATWG, n.d.)

```
<label for="email">Indirizzo email</label>
<input id="email" name="email" type="email">
```

In questo modo il browser associa esplicitamente l’etichetta al campo.

Pseudo-classi

Le pseudo-classi sono parole chiave che si aggiungono a un selettore CSS per individuare un elemento quando si trova in una determinata condizione, posizione o stato, senza dover aggiungere una classe nel codice *HTML*. (W3C CSS Working Group, 2026d)

Le pseudo-classi si scrivono aggiungendo al selettore un singolo carattere di due punti (:) seguito dal nome della pseudo-classe.
Per esempio:

- :hover individua un elemento quando il puntatore si trova sopra di esso;
- :focus individua un elemento che ha ricevuto il focus;
- :checked individua una casella o un pulsante di opzione selezionato;
- :first-child individua un elemento che è il primo figlio del proprio genitore.

A differenza degli pseudo-elementi, che rappresentano una parte dell'elemento o un contenuto generato, le pseudo-classi modificano la corrispondenza di un selettore in base a una determinata condizione. Possono essere dinamiche, come :hover, oppure strutturali, come :first-child. (W3C CSS Working Group, 2026d)

Esempio

```
a:hover {
    color: red;
    text-decoration: underline;
}
```

Il collegamento (che è rappresentato dall'elemento a) diventa rosso e sottolineato quando l'utente vi posiziona sopra il puntatore.

Cascata

Più dichiarazioni CSS possono assegnare valori differenti alla stessa proprietà dello stesso elemento. (W3C CSS Working Group, 2024c)
Per esempio, una regola può assegnare a un paragrafo il colore rosso e un'altra il colore blu. Il browser deve quindi stabilire quale dichiarazione applicare.

Il procedimento attraverso cui vengono risolti questi conflitti prende il nome di **cascata**. (W3C CSS Working Group, 2024c)

Per ogni proprietà di ciascun elemento, il browser individua innanzitutto le dichiarazioni applicabili.
Il selettore deve corrispondere all'elemento e le eventuali condizioni devono essere soddisfatte.
Per esempio, una regola contenuta in @media partecipa alla cascata soltanto quando la condizione è soddisfatta, per esempio quando la finestra ha determinate dimensioni. (W3C CSS Working Group, 2026c)

Successivamente, il browser ordina le dichiarazioni secondo diversi criteri di priorità. In forma semplificata, i principali sono:

Origine e importanza
Il browser considera innanzitutto da dove proviene una dichiarazione: (W3C CSS Working Group, 2024c)

- Dagli stili predefiniti del browser.
- Dalle preferenze o dagli stili dell'utente.
- Dal CSS scritto dall'autore della pagina.

Tra le dichiarazioni normali, quelle dell'autore hanno generalmente la precedenza su quelle dell'utente e del browser.

Una dichiarazione può essere contrassegnata con !important:

```
.avviso {
    color: blue !important;
}
```

L'importanza viene valutata prima della specificità. (W3C CSS Working Group, 2024c)
Una dichiarazione !important può quindi prevalere su una dichiarazione normale associata a un selettore più specifico. Quando vengono confrontate più dichiarazioni !important, continuano però a essere considerati gli altri criteri della cascata.

Livello della cascata
Attraverso @layer, le regole dell'autore possono essere organizzate in livelli con priorità differenti: (W3C CSS Working Group, 2024c)

```
@layer base, componenti;
```

Tra le dichiarazioni normali, quelle contenute nei livelli dichiarati successivamente prevalgono su quelle dei livelli precedenti.
Le dichiarazioni normali non inserite esplicitamente in un livello hanno priorità maggiore rispetto a quelle inserite nei livelli.

Il livello viene confrontato prima della specificità. Un selettore meno specifico collocato in un livello con priorità maggiore può quindi prevalere su un selettore più specifico appartenente a un livello inferiore.

Per le dichiarazioni !important, l'ordine di priorità dei livelli viene invertito.

Specificità del selettore
Quando due dichiarazioni appartengono alla stessa origine, hanno la stessa importanza e si trovano nello stesso livello, il browser confronta la specificità dei selettori. (W3C CSS Working Group, 2024c)

In termini generali:

- I selettori di ID, come #introduzione, hanno una specificità maggiore.
- Seguono classi, attributi e pseudo-classi.
- Seguono i selettori di elementi e gli pseudo-elementi.

Consideriamo questo esempio:

```
#messaggio {
    color: red;
}

.avviso {
    color: blue;
}
```

```
<p id="messaggio" class="avviso">
    Attenzione
</p>
```

Entrambe le regole selezionano il paragrafo.
Poiché hanno la stessa origine e importanza e si trovano nello stesso livello, viene confrontata la loro specificità. Il selettore #messaggio è più specifico di .avviso, quindi il testo risulta rosso. (W3C CSS Working Group, 2024c)

La maggiore specificità dei selettori di ID può rendere più difficile sovrascrivere e riutilizzare gli stili. Per questo motivo, in molti progetti si preferiscono le classi per definire la presentazione dei componenti, mentre gli ID vengono utilizzati soprattutto per identificare elementi univoci, creare destinazioni per frammenti URL e stabilire relazioni tra elementi HTML.

Ordine di apparizione
Se anche la specificità è uguale, prevale la dichiarazione che compare per ultima: (W3C CSS Working Group, 2024c)

```
.avviso {
    color: red;
}

.avviso {
    color: blue;
}
```

In questo caso il testo risulta blu perché le due dichiarazioni hanno la stessa origine, importanza, livello e specificità, ma la seconda compare successivamente nel foglio di stile.

L’ordine di apparizione è quindi l’ultimo criterio di spareggio. Non è corretto affermare che «vince sempre l’ultima regola»: questo accade soltanto quando tutti i criteri precedenti sono equivalenti. (W3C CSS Working Group, 2024c)

Nota:
Quella presentata è una descrizione semplificata dei casi più comuni. La cascata completa considera anche gli stili applicati direttamente mediante l’attributo style, le animazioni, le transizioni, i contesti di incapsulamento e, quando utilizzato, @scope. (W3C CSS Working Group, 2024c)

Layout fluido e media query

Le pagine web possono essere visualizzate in finestre e dispositivi di dimensioni molto differenti. Il CSS permette di adattare la loro presentazione allo spazio disponibile attraverso due strumenti principali: (W3C CSS Working Group, 2026c)

- I layout fluidi
- Le media query

Questi due approcci vengono usati insieme per creare una pagina web il più possibile adatta a diverse dimensioni dello schermo.
Spesso si segue inoltre un approccio **mobile firs**t: si scrivono prima le regole per gli schermi più stretti e si aggiungono progressivamente le configurazioni destinate agli spazi più ampi mediante media query con min-width. (W3C CSS Working Group, 2026c)

Layout fluido
In un layout fluido, le dimensioni degli elementi cambiano gradualmente in relazione allo spazio disponibile. Gli elementi possono inoltre restringersi, allargarsi o ridistribuirsi in base alle regole definite mediante Flexbox o Grid. (W3C CSS Working Group, 2025a, 2025b)
Si utilizzano valori proporzionali e unità flessibili, come percentuali e “fr”, insieme a proprietà come max-width, per impedire che i contenuti diventino eccessivamente grandi. (W3C CSS Working Group, 2024d, 2025b)

Di seguito è riportato un esempio:

```
.pagina {
    width: 90%;
    max-width: 1200px;
    margin: 0 auto;
}

img {
    max-width: 100%;
    height: auto;
}

.galleria {
```

```
    display: grid;
    grid-template-columns: 1fr;
    gap: 20px;
}
```

L'elemento con classe .pagina occupa il 90% della larghezza disponibile, fino a un massimo di 1200px.
La dichiarazione margin: 0 auto lo centra orizzontalmente nel proprio contenitore.

Le immagini possono restringersi fino a non superare la larghezza del contenitore, mentre height: auto ne conserva le proporzioni originali. (W3C CSS Working Group, 2021a)

L'elemento con classe .galleria viene disposto come una griglia a una sola colonna. La colonna ha una dimensione di 1fr e occupa tutto lo spazio disponibile. (W3C CSS Working Group, 2025b)
La proprietà gap stabilisce una distanza di 20px tra gli elementi della griglia.

Media query
Le media query permettono di applicare determinate dichiarazioni *CSS* soltanto quando si verificano specifiche condizioni, per esempio quando la viewport raggiunge una certa larghezza. (W3C CSS Working Group, 2026c)

Una media query usa una **at-rule**, cioè una regola speciale del *CSS* che inizia con il simbolo @ e serve a dare un'istruzione al motore CSS, definendo il contesto in cui le regole contenute al suo interno devono essere applicate. (W3C CSS Working Group, 2026c)

```
@media (min-width: 600px) {
    .galleria {
        grid-template-columns: repeat(2, 1fr);
    }
}

@media (min-width: 1000px) {
    .galleria {
        grid-template-columns: repeat(3, 1fr);
    }
}
```

La at-rule **@media** fa sì che le dichiarazioni contenute al suo interno vengano applicate soltanto quando la condizione indicata risulta vera. (W3C CSS Working Group, 2026c)

In questo caso, min-width: 600px richiede che la larghezza della viewport sia almeno di 600 pixel CSS.

I valori 600px e 1000px sono chiamati **breakpoint**, cioè punti nei quali cambia l'organizzazione del layout. La galleria utilizza una colonna nelle viewport più strette, due

colonne a partire da 600px e tre colonne a partire da 1000px. (W3C CSS Working Group, 2026c)

Quando la viewport raggiunge 1000px, entrambe le media query risultano vere.
Poiché le dichiarazioni hanno la stessa priorità e specificità, prevale quella che compare per ultima e la galleria viene organizzata in tre colonne. Tra un breakpoint e l'altro, le colonne continuano ad adattarsi allo spazio disponibile grazie all'unità flessibile fr. (W3C CSS Working Group, 2024c, 2025b)

Altro esempio:

```
.pagina {
    width: calc(100% - 32px);
    margin: 0 auto;
}

@media (min-width: 768px) {
    .pagina {
        width: 720px;
    }
}

@media (min-width: 1200px) {
    .pagina {
        width: 1120px;
    }
}
```

In questo esempio, nelle viewport inferiori a 768px, la pagina occupa tutta la larghezza disponibile meno 32px, lasciando quindi uno spazio complessivo di 32px ai lati. A partire da 768px assume una larghezza fissa di 720px, mentre da 1200px passa a 1120px.

Esempio di CSS

Si veda un esempio molto semplice di utilizzo del *CSS*.
Prendiamo un codice *HTML*.

```
<!doctype html>
<html lang="it">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <title>Esempio CSS</title>
```

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

<body>
  <main>
    <h1>Titolo del mio sito</h1>
    <p>Questo è un paragrafo di esempio.</p>
  </main>
</body>

</html>
```

L'elemento <link> collega al documento una risorsa esterna dichiarata come foglio di stile. (WHATWG, n.d.)
Infatti Si nota che il documento HTML, mediante l'elemento <link>, collega un foglio di stile esterno chiamato style.css

Vediamo il contenuto di "*style.css*"

```
*,
*::before,
*::after {
  box-sizing: border-box;
}

body {
  margin: 0;
  font-family: system-ui, sans-serif;
  line-height: 1.5;
}

main {
  width: min(100% - 2rem, 60rem);
  margin-inline: auto;
  padding-block: 2rem;
}

h1 {
  color: darkblue;
  font-size: 2rem;
}

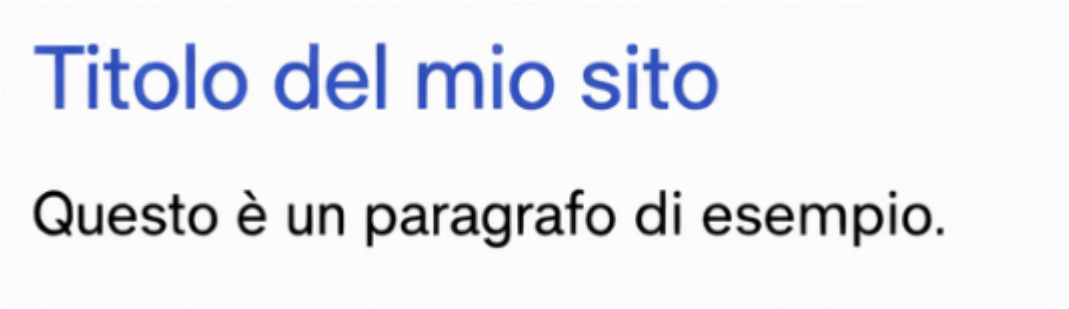
p {
  color: #222;
  font-size: 1rem;
}
```

In questo .css notiamo che sono anche definite le regole di formattazione di due elementi "*h1"* e "*p".*
Questo significa che le regole associate ai selettori h1 e *p* dovranno sottostare alle regole di formattazione imposte dal documento .css.
In questo caso:

- I contenuti dell'elemento h1 saranno di colore darkblue e la grandezza del carattere sarà di 2rem.
- I contenuti dell'elemento p saranno di colore #222 e la grandezza del carattere sarà di 1rem.

Da notare che il browser per visualizzare la pagina deve recuperare il documento *HTML* e, se disponibile, il foglio *CSS* collegato; se il *CSS* non viene recuperato, il documento *HTML* può comunque essere visualizzato, semplicemente senza gli stili definiti in quel foglio. (WHATWG, n.d.)

Di seguito è mostrata la pagina web prodotta utilizzando questi due documenti.



Materiale di supporto

Esercizi

Separare struttura e presentazione
Considera questo documento HTML:

```
<main>
    <h1>Portfolio fotografico</h1>
    <p>Una selezione dei miei lavori.</p>
    <a href="/portfolio">Guarda i progetti</a>
</main>
```

Il progettista vuole ottenere queste caratteristiche:

- Il titolo deve essere blu scuro e grande.
- Il paragrafo deve essere grigio.
- Il collegamento deve apparire come un pulsante.
- Tutte le pagine del sito devono utilizzare lo stesso sistema visivo.

Rispondi alle seguenti domande:

- Spiega perché è più appropriato intervenire con CSS anziché modificare la struttura HTML per ottenere questi risultati.
- Scrivi un possibile foglio CSS che realizzi le caratteristiche richieste.
- Per ogni regola individua il selettore, le proprietà e i relativi valori.
- Spiega perché cambiare graficamente il collegamento non modifica il suo significato di collegamento.
- Indica come collegheresti un file style.css al documento HTML.
- Spiega perché utilizzare un foglio di stile esterno può essere vantaggioso in un sito composto da cinquanta pagine.
- Immagina di voler cambiare il colore principale dell'intero sito. Spiega perché la separazione tra HTML e CSS rende più semplice questa operazione.

Quanto è grande davvero il box?
Un elemento utilizza questo CSS:

```
.card {
    width: 300px;
    padding: 20px;
    border: 5px solid black;
    margin: 30px;
}
```

Per impostazione predefinita viene utilizzato:

```
box-sizing: content-box;
```

- Indica la larghezza della content area.

- Calcola la larghezza complessiva occupata da contenuto, padding e bordo.
- Calcola lo spazio complessivo considerando anche i margini laterali.
- Spiega la differenza tra padding e margin.
- Indica quale delle due aree è normalmente coperta dallo sfondo dell'elemento.
- Riscrivi la regola utilizzando box-sizing: border-box.
- Spiega cosa cambia nel significato di width: 300px utilizzando border-box.
- Un contenitore è largo 800 px e un elemento figlio possiede width: 50%. Spiega quale larghezza avrà la sua content area, assumendo che il containing block corrisponda a quel contenitore e che non intervengano altri vincoli.
- Aggiungi max-width: 300px e spiega cosa succederebbe se il contenitore diventasse largo 1000 px.

Scegliere il selettore corretto

Considera questo HTML:

```
<form>
    <input type="text" placeholder="Nome">
    <input type="email" placeholder="Email">
    <button disabled>Invia</button>
</form>

<a href="manuale.pdf">Scarica il manuale</a>
<a href="https://example.com">Visita il sito</a>
```

Scrivi un selettore CSS appropriato per:

- Selezionare soltanto il campo email.
- Selezionare tutti gli elementi che possiedono l'attributo disabled.
- Selezionare soltanto i collegamenti che conducono a un file PDF.
- Selezionare soltanto i collegamenti il cui indirizzo inizia con https.
- Modificare il colore del placeholder del campo email.
- Cambiare stile a un collegamento quando l'utente vi passa sopra con il mouse.
- Spiega la differenza tra selettore di attributo, pseudo-classe e pseudo-elemento.
- Spiega perché :hover descrive uno stato dell'elemento, mentre ::placeholder individua una parte specifica della sua rappresentazione.
- Scrivi infine un selettore che individui il placeholder del campo email soltanto quando questo si trova all'interno di un <form>.

Classe o ID?

Devi progettare una pagina contenente:

- Dodici card di progetto.
- Tre pulsanti principali.
- Uno di questi pulsanti deve avere una variante evidenziata.
- Un'unica intestazione principale.
- Una sezione con identificatore contatti, raggiungibile tramite un link interno.

Progetta la struttura degli identificatori e delle classi.

- Indica quali elementi gestiresti attraverso classi.
- Indica quale elemento potrebbe essere identificato tramite ID.

- Proponi una classe riutilizzabile per tutte le card.
- Proponi una classe generale per i pulsanti e una seconda classe per la variante evidenziata.
- Scrivi un esempio di elemento HTML che utilizzi entrambe le classi.
- Spiega perché è corretto assegnare la stessa classe a più elementi.
- Spiega perché invece un valore id deve essere unico nel documento.
- Scrivi il collegamento che permette di raggiungere la sezione contatti.
- Spiega perché, secondo la dispensa, le classi sono generalmente più adatte degli ID per definire componenti visivi riutilizzabili.

Quale regola vince?

Considera questo codice:

```
<p id="messaggio" class="avviso">
    Attenzione
</p>
```

e questo CSS:

```
p {
    color: green;
}

.avviso {
    color: blue;
}

#messaggio {
    color: red;
}

.avviso {
    color: orange;
}
```

- Indica quali regole si applicano al paragrafo.
- Determina quale colore verrà visualizzato.
- Spiega perché il selettore #messaggio prevale sui selettori .avviso.
- Spiega perché non è corretto dire semplicemente «vince sempre l'ultima regola».
- Rimuovi la regola con #messaggio e determina quale colore verrà applicato.
- Spiega perché in questo secondo caso prevale orange.
- Aggiungi alla prima regola .avviso la dichiarazione color: blue !important; e determina nuovamente il risultato.
- Spiega il ruolo di importanza, specificità e ordine di apparizione nella cascata.
- Spiega perché un utilizzo eccessivo di ID e !important può rendere più difficile la manutenzione degli stili.

Progettare una galleria responsive

Devi realizzare una galleria fotografica che funzioni bene su smartphone, tablet e desktop. Le richieste sono:

- Sugli schermi piccoli deve essere presente una sola colonna.
- Da 600 px devono comparire due colonne.
- Da 1000 px devono comparire tre colonne.
- La pagina non deve mai superare 1200 px di larghezza.
- Le immagini non devono uscire dal proprio contenitore.
- Il layout deve continuare ad adattarsi fluidamente anche tra un breakpoint e l'altro.

Scrivi un possibile CSS utilizzando Grid, unità relative, max-width e media query.
Successivamente:

- Spiega perché la soluzione può essere definita responsive.
- Individua quali parti della soluzione costituiscono un layout fluido.
- Individua i breakpoint.
- Spiega cosa accade a una viewport di 500 px.
- Spiega cosa accade a una viewport di 800 px.
- Spiega cosa accade a una viewport di 1200 px.
- A 1200 px risultano vere entrambe le media query. Spiega perché viene comunque utilizzata la configurazione a tre colonne.
- Spiega il significato dell'unità fr nella griglia.
- Spiega perché max-width: 100% e height: auto possono essere utili per le immagini.
- Spiega perché questa impostazione può essere considerata mobile first.
- Concludi spiegando la differenza tra adattamento continuo del layout fluido e cambiamento condizionale introdotto dalle media query.

Bibliografia

WHATWG. (n.d.). *HTML Standard*.

W3C CSS Working Group. (2021a). *CSS Box Sizing Module Level 3*. World Wide Web Consortium.

W3C CSS Working Group. (2021b). *CSS Syntax Module Level 3*. World Wide Web Consortium.

W3C CSS Working Group. (2024a). *CSS Backgrounds and Borders Module Level 3*. World Wide Web Consortium.

W3C CSS Working Group. (2024b). *CSS Box Model Module Level 3*. World Wide Web Consortium.

W3C CSS Working Group. (2024c). *CSS Cascading and Inheritance Level 6*. World Wide Web Consortium.

W3C CSS Working Group. (2024d). *CSS Values and Units Module Level 4*. World Wide Web Consortium.

W3C CSS Working Group. (2025a). *CSS Flexible Box Layout Module Level 1*. World Wide Web Consortium.

W3C CSS Working Group. (2025b). *CSS Grid Layout Module Level 2*. World Wide Web Consortium.

W3C CSS Working Group. (2025c). *CSS Pseudo-Elements Module Level 4*. World Wide Web Consortium.

W3C CSS Working Group. (2026a). *CSS Display Module Level 3*. World Wide Web Consortium.

W3C CSS Working Group. (2026b). *CSS Snapshot 2026*. World Wide Web Consortium.
W3C CSS Working Group. (2026c). *Media Queries Level 4*. World Wide Web Consortium.
W3C CSS Working Group. (2026d). *Selectors Level 4*. World Wide Web Consortium.

Approfondimenti

- Jon Duckett, HTML & CSS. Progettare e costruire siti web, Apogeo, 2017 — Introduce CSS attraverso un approccio fortemente visuale, approfondendo selettori, colori, tipografia, box model, dimensionamento, layout e progettazione delle pagine. È particolarmente adatto per consolidare in modo pratico i concetti fondamentali presentati nella dispensa e comprenderne l'applicazione all'interno di un progetto web completo.
- Eric A. Meyer ed Estelle Weyl, CSS: The Definitive Guide, 5ª ed., O'Reilly Media, 2025, edizione in italiano — Costituisce l'approfondimento tecnico più completo della selezione. Analizza in dettaglio la struttura delle regole, i selettori, le pseudo-classi. pseudo-elementi, cascata, specificità, box model, layout, Flexbox, Grid e le funzionalità del CSS moderno. È particolarmente indicato per passare dalla conoscenza introduttiva della dispensa a una comprensione rigorosa del funzionamento del linguaggio.
- Gianluca Troiani, Responsive Web Design, Apogeo, 2013 — Approfondisce specificamente la progettazione di interfacce capaci di adattarsi a dispositivi e viewport differenti, affrontando layout fluidi, immagini responsive, CSS3, media query, breakpoint e strumenti di test. È quindi direttamente collegato alla parte della dispensa dedicata a layout fluido, media query e responsive design.
- Peter Gasston, Sviluppare applicazioni web multi-device. Con HTML5, CSS3 e JavaScript, Apogeo, 2013 — Esamina la realizzazione di interfacce utilizzabili su dispositivi differenti, approfondendo CSS3, organizzazione fluida dei contenuti e tecniche responsive. È utile per comprendere come le regole CSS si inseriscano in una strategia più ampia di progettazione web multi-device.
- Daniele Bochicchio e Stefano Mostarda, HTML 5 con CSS e JavaScript, 2ª ed., Hoepli, 2018 — Fornisce una trattazione pratica delle tecnologie del Web e dedica specifiche sezioni all'introduzione ai CSS, ai layout, agli effetti CSS3 e alla gestione tipografica. È utile soprattutto per osservare come HTML, CSS e JavaScript collaborino nella costruzione di un'interfaccia web completa.

Nota editoriale

Per la realizzazione di questi materiali ho utilizzato strumenti di intelligenza artificiale generativa come supporto alla scrittura, in particolare per migliorare la forma del testo, riorganizzare i contenuti, correggere la formulazione e velocizzare alcune attività redazionali.

Lavorando autonomamente alla produzione di questi materiali, cerco infatti di automatizzare tutte le attività che possono esserlo, così da poter dedicare più tempo alla ricerca, alla progettazione e allo sviluppo dei contenuti.

L'intelligenza artificiale non determina però i contenuti dell'opera: la scelta degli argomenti, la struttura, le idee, le interpretazioni, gli esempi e l'impostazione didattica sono elaborati da me. L'IA viene quindi utilizzata principalmente come strumento di supporto alla produzione e alla revisione formale, mentre la responsabilità autoriale e progettuale dei contenuti rimane mia.

Salvo diversa indicazione, questo materiale è distribuito con licenza Creative Commons Attribuzione – Non commerciale – Condividi allo stesso modo 4.0 Internazionale (CC BY-NC-SA 4.0).

È quindi possibile condividere, ridistribuire, modificare e creare opere derivate a partire dal materiale, a condizione di attribuirne correttamente la paternità, di non utilizzarlo per scopi commerciali e di distribuire eventuali versioni modificate o derivate con la stessa licenza.

