
HTML
CSS

HTML

- HTML è un **linguaggio di markup**
- La sua natura è racchiusa in "marcatori" chiamati **tag** o **elementi**:

```
<p>Marte è il quarto pianeta del sistema solare</p>
```

- I tag sono composti da un'**apertura** (“ "), un **contenuto** (“Marte...") e una **chiusura** (“ ")
- Esistono anche tag **senza** contenuto e chiusura:

```
<img src="...">
```

Tipologie di elementi

- Gli elementi si possono dividere in **due** macro categorie
- Gli elementi **BLOCK**:
  - Occupano **tutto** lo spazio orizzontale del loro contenitore
  - Forzano i successivi tag ad accodarsi **verticalmente**
  - Supportano tutte le proprietà del **box-model** (vedremo in seguito)
- Gli elementi **INLINE**:
  - Occupano solo lo spazio necessario al loro **contenuto**
  - Permettono ad altri elementi inline di accodarsi **orizzontalmente**
  - Non permettono di alterare la loro **dimensione** (width/height)

Qualche esempio

- Elementi BLOCK:

```
<h1>...</h1>
<p>...</p>
<div>...</div>
<table>...</table>
<ul>...</ul>
<li>...</li>
```

- Elementi INLINE:

```
<a>...</a>
<img>
<b>...</b>
<strong>...</strong>
<span>...</span>
```

Nesting

- Ovviamente gli elementi possono essere **annidati** fra loro

```
<div>
  <p>
    <a href="...">Marte</a> è il quarto pianeta
  </p>
</div>
```

- Una buona regola è **evitare** di annidare elementi BLOCK in elementi INLINE. Meglio affidarsi al CSS per alterarne il comportamento.
- Attenzione al signor **<p>**, vi imbroglierà

```
<p>
  <div>...</div>
</p>
```

La pagina HTML5 standard

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

  <head>
    <title>Marte</title>
    <meta charset="utf-8">
  </head>

  <body>
    ...
  </body>

</html>
```

Gli elementi principali (1/2)

- Le intestazioni: **<h1-6>** (heading)
- Il paragrafo di testo: **<p>** (paragraph)
- Il link: **<a>** (anchor)
- L'immagine: **<img>** (image)
- L'elemento BLOCK per eccellenza: ** ** (division)
- L'elemento INLINE per eccellenza: ****
- Il ritorno a capo: **
** (break)

Gli elementi principali (2/2)

- Gli elementi di stile: **<b>** (bold), **<i>** (italic), **<u>** (underline), **<sup>** (superscript), **<sub>** (subscript), **<small>**
- Le liste: **<ul>** (unordered list), **<ol>** (ordered list), **<li>** (list item)
- La tabella: **<table>**, **<thead>** (table head), **<tbody>** (table body), **<tr>** (table row), **<td>** (table data), **<tfoot>** (table footer)
- Il form: **<form>**, **<label>**, **<input>**, **<textarea>**, **<button>**, **<select>**, **<option>**
- La citazione: **<blockquote>**, **<cite>**
- La riga separatoria: **<hr>** (horizontal rule)

Lista nera

Esistono molti elementi usati in passato che oggi sono **deprecati**:

- Tutti gli elementi di stile (** **, **<font>**, **<strike>** ecc)
- Tutti gli attributi di stile (**align=""**, **width=""**, **height=""** ecc)
- **<frameset>** e **<frame>** (da non confondere con <iframe>)
- Trovate la lista completa su developer.mozilla.org

CSS (Cascading Style Sheets)

- Lo stile non è il lavoro dell'HTML
- La natura del CSS è racchiusa in **regole**:

```
h3 {
  color: red;
  margin: 25px;
}
```

- Le regole sono formate da un **selettore** ("h3") e da un elenco di **proprietà** ("color" e "margin") e **valore** ("red" e "25px")

3 modi per includere il CSS

- INLINE nell'HTML: da **evitare**

```
<h3 style="with: 200px;">...</h3>
```

- EMBEDDED STYLE nell'HTML: da **evitare**

```
<style>
  h3 {
    with: 200px;
  }
</style>
```

- Tag LINK nell'HTML e file CSS esterno: **OK**

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

```
h3 {
  with: 200px;
}
```

Selettori principali

- Selettore di **elementi**

```
<h3>...</h3>
```

```
h3 { ... }
```

- Selettore di **ID**

```
<h3 id="title">...</h3>
```

```
#title { ... }
```

- Selettore di **classe**

```
<h3 class="title">...</h3>
```

```
.title { ... }
```

Selettori secondari (1/2)

- Selettore **child**

```
<p>
  <a>...</a>
</p>
```

```
p > a { ... }
```

- Selettore **descendant**

```
<p>
  <a>...</a>
  <span>
    <a>...</a>
  </span>
</p>
```

```
p a { ... }
```

- Selettore **adjacent sibling**

```
<p>...</p>
<a>...</a>
```

```
p + a { ... }
```

Selettori secondari (2/2)

- Selettore **general sibling**

```
<p>...</p>
<span>...</span>
<a>...</a>
```

```
p ~ a { ... }
```

- Selettore d'**attributo**

```
<a target="_self">...</a>
<a target="_blank">...</a>
```

```
a[target] { ... }
a[target="_blank"] { ... }
```

- Selettore **star**

```
p > * { ... }
```

- **Grouping** di selettori

```
a, p, h3 { ... }
```

Pseudo selettori (1/2)

- Pseudo selettore **focus**

```
button:focus { ... }
```

- Pseudo selettore **hover**

```
button:hover { ... }
```

- Pseudo selettore **not**

```
button:not(.warning) { ... }
```

- Pseudo selettore **checked**

```
input:checked { ... }
```

Pseudo selettori (2/2)

- Pseudo selettore **first-child**

```
li:first-child { ... }
```

- Pseudo selettore **last-child**

```
li:last-child { ... }
```

- Pseudo selettore **nth-child**

```
li:nth-child(even) { ... }
li:nth-child(odd) { ... }
li:nth-child(3) { ... }
```

- Trovate la lista completa su developer.mozilla.org

Pseudo elementi

- Pseudo elemento **before**

```
p::before {
  color: red;
  content: 'Iniziamo con Ciao!';
}
```

- Pseudo elemento **after**

```
p::after {
  color: red;
  content: 'Finiamo con Ciao!';
}
```

Specificità dei selettori

- Un concetto avanzato dei selettori è la loro **specificità** ovvero quando un selettore ha priorità su un altro, sovrascrivendolo.
- I selettori seguono la gerarchia "id" > "classe" > "elemento"
- Possiamo vedere degli esempi e **misurare** la specificità di un selettore su specificity.keegan.st
- A parità di selettore, verrà applicato l'**ultimo** caricato in pagina

Il selettore più affidabile (1/2)

- La **classe** permette di **riutilizzare** una regola CSS e di creare selettori **mirati** e **compatti**
- Una buona pratica è quella di creare una classe **principale** per l'elemento padre e utilizzarla come **prefisso** per il suo contenuto
- Questo **previene** selettori annidati, poco chiari e complessi

Il selettore più affidabile (2/2)

```
<section class="product">
  <h3 class="product-header">...</h3>

  <p class="product-body">
    <a href="..." class="product-link">...</a>
  </p>
</section>
```

```
section { ... }

section h3 { ... }

section p { ... }

section p a { ... }
```

```
.product { ... }

.product-header { ... }

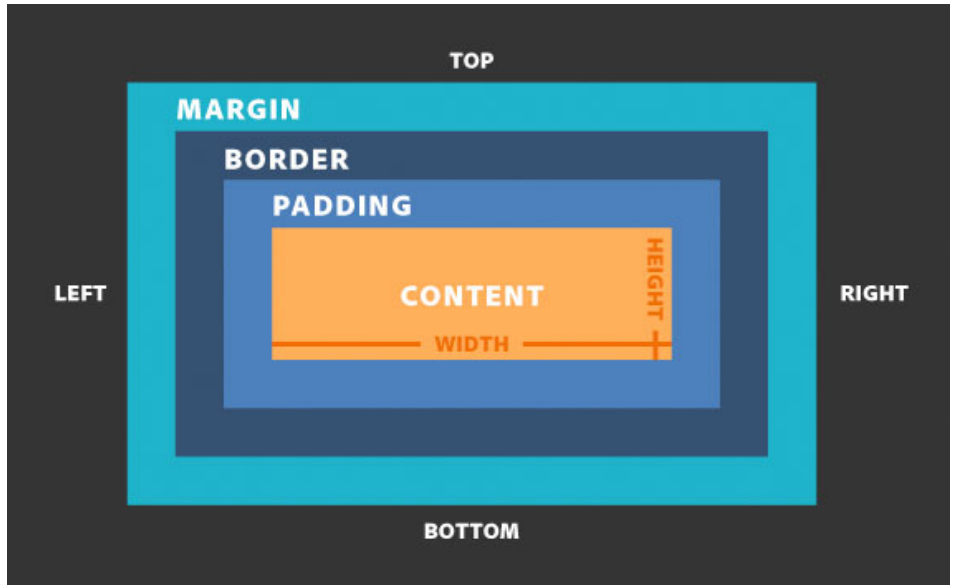
.product-body { ... }

.product-link { ... }
```

Il box model (1/4)

Ogni elemento renderizzato in pagina sarà presentato come un **box**.
Con box model si fa riferimento a un **set specifico** di proprietà CSS:

- width
- height
- padding
- border
- margin



Il box model (2/4)

- Il box model **NON** è uguale per tutti gli elementi
- Gli elementi BLOCK possiedono **tutte** le proprietà del box model
- Gli elementi INLINE **non** permettono la modifica di:
  - width
  - height
  - margin-top
  - margin-bottom

Il box model (3/4)

- La tipologia degli elementi può essere alterata con la proprietà **display**

```
h3 {
  display: inline;
}
```

- I **valori** principali sono:
  - inline
  - block
  - flex
  - grid
  - inline-*
  - none

@visibile in demo

Il box model (4/4)

- Altre proprietà CSS legate al box model sono:

```
.box {
  box-sizing: content-box | border-box;
  max-height: 100px;
  min-height: 100px;
  max-width: 100px;
  min-width: 100px;
  opacity: 0-1;
  overflow: auto | scroll | hidden | visible;
  overflow-x: auto | scroll | hidden | visible;
  overflow-y: auto | scroll | hidden | visible;
  vertical-align: top | middle | bottom | baseline;
  visibility: visible | hidden | collapse;
}
```

Posizionamento

- La posizione degli elementi può essere alterata con la proprietà **position**

```
h3 {
  position: absolute;
}
```

- I **valori** possibili sono:
  - static
  - relative
  - absolute
  - fixed
  - sticky

Si accompagnano con le **proprietà**:

- top | bottom
- left | right

- Un'altra proprietà dedicata al posizionamento è **float**

@visibile in demo

Background

Il background rimane una proprietà ever-green

```
.product {
  background-color: #000;
  background-image: url("...");
  background-origin: padding-box | border-box | content-box;
  background-position: top | bottom | center | left | right;
  background-repeat: repeat | repeat-x | repeat-y | no-repeat;
  background-size: % | auto | cover | contain;
}
```

@visibile in demo

Colori

Esistono diversi formati con cui poter specificare un colore:

- ```
  .color-by-name { background-color: red; }
  ```
- ```
  .color-by-short-hex { background-color: #F00; }
  ```
- ```
  .color-by-long-hex { background-color: #FF0000; }
  ```
- ```
  .color-by-short-hex-a { background-color: #F00A; }
  ```
- ```
  .color-by-rgb { background-color: rgb(255, 0, 0); }
  ```
- ```
  .color-by-rgba { background-color: rgb(255, 0, 0, 100); }
  ```
- ```
  .color-by-hsl { background-color: hsl(0, 100%, 50%); }
  ```
  ```

Testo

- Il CSS offre un numero elevato di proprietà per customizzare il testo

```
.text {
  color: #F00;
  font-family: Arial, sans-serif;
  font-size: 18px;
  font-style: normal | italic | oblique;
  font-variant: normal | small-caps;
  font-weight: lighter | normal | bold | bolder | 100-900;
  letter-spacing: 1px;
  line-height: 20px;
  text-align: left | center | right;
  text-decoration: none | underline | overline | line-through;
  text-overflow: clip | ellipsis;
  white-space: normal | nowrap | pre | pre-line;
  word-break: normal | break-word | break-all;
  word-spacing: 1px;
}
```

Testo relativo

- Una funzionalità interessante dei testi è la **dimensione relativa (em)** che permette di creare layout con testi facilmente ridimensionabili

```
body {
  font-size: 14px;
}
.product-header {
  font-size: 28px;
}
.product-body {
  font-size: 21px;
}
.product-footer {
  font-size: 12px;
}
```

```
body {
  font-size: 14px;
}
.product-header {
  font-size: 2em;
}
.product-body {
  font-size: 1.5em;
}
.product-footer {
  font-size: 1em;
}
```

@visibile in demo

Liste e Tabelle

- Oggi giorno, il CSS utilizzato per liste e tabelle mira semplicemente a "resettarne" lo stile e customizzarlo

```css
ul {
  list-style: none;
  padding: 0;
}
```

```css
table {
  border-collapse: collapse;
  table-layout: fixed;
}
```

FlexBox e Grid (1/4)

- Una delle difficoltà avute in passato con i CSS fu la creazione dei **layout** (siti tabella?)
- Si doveva ricorrere a molti "trick" per affiancare delle colonne come l'abuso di **tabelle** o del **float**
- In questi ultimi anni sono state introdotte due nuove soluzioni a questo problema: il **FlexBox** e il **Grid**

```
.layout {
  display: flex;
}
```

```
.layout {
  display: grid;
}
```

FlexBox e Grid (2/4)

- La differenza tra i due risiede nel tipo di **layout target**:
  - il FlexBox è pensato per layout **mono-dimensionali** organizzati in righe **o** colonne
  - il Grid è pensato per layout **bi-dimensionali** organizzati in righe **e** colonne
- Possono essere **combinati** senza problemi per creare layout avanzati
- Non è mai così semplice (Browser Support)
  - FlexBox a partire da **IE10** con diversi handicap, OK da **Edge 12**
  - Grid a partire da **IE10** con diversi handicap, OK da **Edge 16**

FlexBox e Grid (3/4)

- Le proprietà principali del FlexBox:

```
.layout {
  align-items: center | flex-start | flex-end | stretch;
  display: flex; /* unica obbligatoria */
  flex-direction: row | column;
  flex-wrap: wrap | nowrap;
  justify-content: center | flex-start | flex-end | stretch;
}
```

- Le proprietà principali degli elementi contenuti in un FlexBox:

```
.layout-child {
  flex-basis: 100px;
  flex-grow: 1;
  flex-shrink: 0;
}
```

- Trovate tutte le funzionalità su [css-tricks.com]

FlexBox e Grid (4/4)

- Le proprietà principali del Grid:

```
.layout {
  display: grid;
  grid-gap: 10px 20px; /* rows e columns */
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: 25% 50% 25%;
}
```

- Trovate tutte le funzionalità su [css-tricks.com]

@visibile in demo

Media query e layout responsive

- Le media query sono strumenti utili a realizzare **layout responsive** ovvero adatti sia a schermi grandi (desktop) che piccoli (mobile).
- La struttura di una media query è composta dal prefisso **@media** seguito dal dispositivo **target** e dalla **condizione** da rispettare

```
@media all and (min-width: 800px) {
  .layout-column {
    width: 50%;
  }
}
```

- I **target** sono "all", "print", "screen" e "speech"
- Le **condizioni** disponibili sono "min-width", "max-width", "min-height" e "max-height"

@visibile in demo

Best Practices

1. **Non** utilizzare il **CSS inline**
2. **Non** utilizzare il **CSS inline**
3. **Utilizzare le classi** come selettore e strutturarle col prefisso "padre-figlio"
4. **Limitare** al minimo il **nesting** nei selettori
5. **Non** utilizzare **!important**
6. Scrivere le proprietà CSS in **ordine alfabetico** (abitudine)
7. **Evitare** di inserire elementi BLOCK **dentro** elementi INLINE nell'HTML

Developer Tools

- Un fedele alleato degli svilupptori front-end è la **console** di sviluppo fornita da tutti i Browser
- In particolare, il tab "Elements" (es. di Chrome) che ci mostra l'**alberatura** renderizzata secondo le regole del browser e gli **stili calcolati**
- Possibile anche **simulare** gli pseudo selettori o le media query