

# Corso Front End Developer

## CSS

Emanuele Galli

[www.linkedin.com/in/egalli/](https://www.linkedin.com/in/egalli/)

# CSS: Cascading Style Sheets

- 1996 World Wide Web Consortium (W3C), versione corrente: CSS3
- Separazione tra contenuto e presentazione in un documento HTML
- Lo stile è definito da regole
- Ogni **regola** è strutturata in
  - **Selettore**: a quali elementi si applica la regola
  - **Dichiarazioni**: come devono essere “stilati” gli elementi



# HTML e CSS




- Si possono “stilare” elementi di un documento HTML

- Nella HEAD

- Definendo inline lo stile in un **elemento style** (sconsigliato in produzione)
- Definendo un collegamento a un file CSS esterno
  - via un **elemento link**
  - via import all'interno di un **elemento style**

- Nel BODY

- Nello specifico elemento usando l'**attributo style** (sconsigliato in produzione)




```
<head>
<!-- -->
<style>input {color: red;}</style>
</head>
```




```
input {color: red;}
```

css/s3.css



```
<link rel="stylesheet"
type="text/css"
href="css/s3.css"/>
```



```
<style type="text/css">
@import url(css/s3.css);
</style>
```

# Cascading 🗨️🗨️

Più regole per lo stesso selettore, vince l'ultima

```
p {  
  font-family: Arial;  
  font-size: 12pt;  
}  
  
p {  
  color: red;  
  font-size: 11pt;  
}  
  
p {  
  margin-left: 15pt;  
  color: green;  
}
```



```
p {  
  font-family: Arial;  
  font-size: 11pt;  
  margin-left: 15pt;  
  color: green;  
}
```

# Selettori



```
p { ... }  
.className { ... }  
#idName { ... }  
[type=text]  
:first-child { ... }  
::before
```



```
div>span { ... }  
div span { ... }  
h1 + p { ... }
```

```
h1, h2, h3 { ... }  
input:hover { ... }  
p.className { ... }
```



- Selezione degli elementi nella pagina a cui applicare la regola:
  - Tipo dell'elemento
  - Classe, attributo class
  - Identificatore, attributo id
  - Attributo
  - Pseudo classe (hover, checked, nth-child(), ...), anchor → link, visited
  - Pseudo elemento (before, after, selection, first-letter, ...)
  - Discendenza diretta
  - Discendenza generica
  - Stesso livello, elemento successivo
- Più selettori possono essere associati a una regola
- I selettori possono essere combinati
- Le regole si applicano in cascata
- In generale, in caso di conflitto vince la regola più specifica


# Uso di id e class


```
div.highlight {  
  text-align: center;  
  color: red;  
}  
  
span#important {  
  background-color: yellow;  
}
```

```
<div class="highlight">  
  <h1>Something important</h1>  
  <h2>very important!</h2>  
</div>
```

```
<div>  
  <p>  
    A few words with  
    <span id="important">something</span>  
    more relevant  
  </p>  
</div>
```


# Selettori – esempi

```
[type=text] {  
    background-color: olive;  
}  
  
[type=number] {  
    background-color: yellow;  
}  
  
input:hover {   
    color: blue;  
}
```



```
<input name="firstname" type="text">  
<input name="lastname" type="text">  
<input name="age" type="number">
```

```
div span {  
    background-color: yellow;  
}  
  
div>span {  
    font-weight: bold;  
}
```



```
<div>  
    <span>A</span> <span>B</span>  
    <p>  
        <span>C</span> <span>D</span>  
    </p>  
</div>  
<p>  
    <span>E</span> <span>F</span>  
</p>
```

# Proprietà

- Alcune tra le proprietà più usate in CSS:
  - **background**: sfondo di un elemento
    - **background-color**: (yellow, #129921) ...
  - **border**: il bordo di un elemento (border: 1px solid black;)
    - **border-width**, **border-color**, **border-collapse**, ...
  - **color**: colore del testo nell'elemento
  - **font**: proprietà del carattere per il testo nell'elemento
    - **font-size** (80%, 1.2em, 18px), **font-family** (Arial, sans-serif), **font-style** (italic), **font-weight** (bold)
  - **margin** e **padding**: spazio attorno all'elemento (esterno e interno ai bordi)
  - **text-align** (center, justify): allineamento del testo
  - **text-transform**: (uppercase, capitalize)
  - **width**, **height**: dimensioni, quando applicabili



# Esempio: tabella con CSS

```
<table>
  <tr>
    <th>Left</th>
    <th>Center</th>
    <th>Right</th>
  </tr>
  <tr>
    <td>7</td>
    <td>8</td>
    <td>9</td>
  </tr>
  <tr>
    <td>5</td>
    <td>6</td>
    <td>7</td>
  </tr>
  <tr>
    <td>1</td>
    <td>2</td>
    <td>3</td>
  </tr>
</table>
```

```
table {
  border: 2px solid black;
  border-collapse: collapse;
  Width: 50%;
}

td, th {
  border: 1px solid red;
  padding: 3px;
  text-align: center;
}



th {
  background-color: lightblue;
}

td {
  background-color: lightgreen;
}
```

# Box model

- Ogni elemento è visto come una scatola
  - **margin**: spazio bianco esterno all'elemento
  - **border**: bordo (invisibile per default)
  - **padding**: spazio bianco interno attorno al contenuto
  - **content**: testo o elementi contenuti nell'elemento


# Tools

- Preprocessore SCSS
  - <https://sass-lang.com/> (SASS è il nome originale)
  - Variabili
  - Nesting
  - Moduli, mixin, gerarchie
  - Operatori
- Minify  

# Bootstrap

- Framework CSS per lo sviluppo web front-end (più modulo JavaScript opzionale)
- Progetto interno di Twitter, 2011
- Download da <https://getbootstrap.com/>  
`<link rel="stylesheet" href="css/bootstrap.min.css">`
- CDN da <https://www.bootstrapcdn.com/>  
`<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">`

# Setup

- Assicurarsi che il browser interpreti la pagina come HTML5
  - `<!doctype html>`
- Head
  - Inserire i seguenti **meta**
    - `<meta charset="utf-8">`
    - `<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">` 
  - Inserire il **link** a Bootstrap

# Container

- Due tipi di container
  - **container**, lunghezza fissa per ogni breakpoint
  - **container-fluid**, è sempre il 100% del viewport

```
<div class="container">  
  <h1>Hello from Bootstrap</h1>  
</div>
```

```
<div class="container-fluid">  
  <h1>Hello from Bootstrap</h1>  
</div>
```



# Grid

- All'interno di un container, gli elementi sono organizzati in righe e colonne
- Un div di classe **row** per ogni riga
- Un div di classe **col** per ogni cella, implicitamente tutte della stessa dimensione

```
<div class="container-fluid">  
  <div class="row">  
    <div class="col">1/1</div>  
    <div class="col">2/1</div>  
    <div class="col">3/1</div>  
  </div>  
  <div class="row">  
    <div class="col">1/2</div>  
    <div class="col">2/2</div>  
    <div class="col">3/2</div>  
  </div>  
</div>
```

# Breakpoint





- La dimensione del viewport viene categorizzata in **breakpoint**  
extralarge (**xl**), large (**lg**), medium (**md**), small (**sm**)
- Ogni col può avere una dimensione in dodicesimi

```
<div class="row">  
  <div class="col-sm-2 col-md-3 col-lg-5 col-xl-1">1</div>  
  <div class="col-sm-4 col-md-3 col-lg-1 col-xl-5">2</div>  
  <div class="col-sm-4 col-md-3 col-lg-1 col-xl-5">3</div>  
  <div class="col-sm-2 col-md-3 col-lg-5 col-xl-1">4</div>  
</div>
```





# Table

- Per stilare un elemento table lo si mette in un container, gli si applica la classe table e ...
  - table-borderless
  - table-dark
  - table-striped 
  - table-bordered
  - table-hover 
  - table-sm 
- Classi per thead 
  - thead-dark, thead-light
- Classi per table, th, tr, td
  - table-success, table-danger, table-info, table-warning, ...

```
<table>
  <thead>
    <tr>
      <th scope="row">\</th>
      <th scope="col">Left</th>
      <th scope="col">Right</th>
    </tr>
  </thead>
  <tr>
    <th scope="row">Top</th>
    <td>X</td>
    <td>Y</td>
  </tr>
  <tr>
    <th scope="row">Low</th>
    <td>1</td>
    <td>2</td>
  </tr>
</table>
```