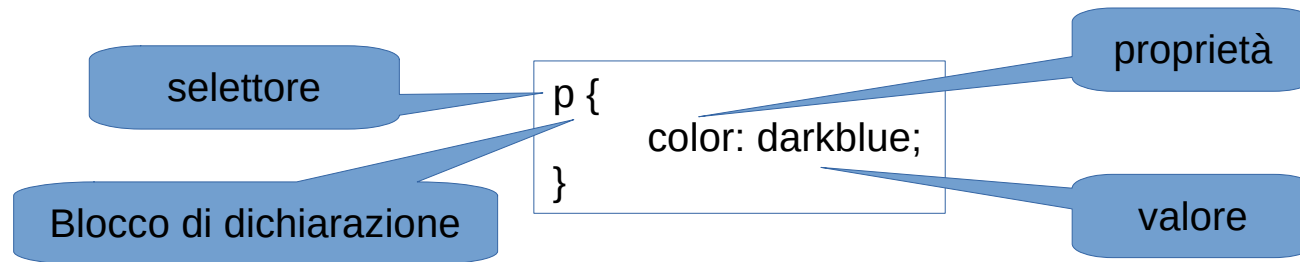


CSS

- Regole, selettori, dichiarazioni di stile
- Integrazione con HTML
- SASS
- Progetto di riferimento
 - <https://github.com/egalli64/nesp> (*modulo 2*)
 - Node.js
 - VS Code

CSS: Cascading Style Sheets

- 1996 World Wide Web Consortium (W3C), versione corrente: CSS3
- Separazione tra struttura + contenuto e presentazione in un documento HTML
 - Definisce come gli elementi devono essere rappresentati (“stilati”)
- Lo **stile** è definito da una o più regole
- Ogni **regola** è strutturata in
 - **Selettore**: indica a quali elementi si applica la regola
 - Blocco che contiene una o più **dichiarazioni** sullo “stile” degli elementi (colori, formattazione, dimensioni, ...)
 - Coppie nella forma *proprietà: valore*;



HTML e CSS

- Si può “stilare” un documento HTML definendo

- Nel BODY

- *Inline*, in un elemento usando l'attributo style
(*sconsigliato – niente esempi*)

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

- Nella **HEAD**

- Un elemento style (sconsigliato in produzione)
 - Riferendosi a un **file CSS esterno**
 - via un elemento **link**
 - via CSS import all'interno di un elemento **style**

```
/* a comment */
input {color: red;}
```

css/s03.css

```
<link rel="stylesheet"
      type="text/css"
      href="css/s03.css">
```

- Uso di LINK per risorse esterne al browser

- Esempio: Google fonts

- <https://fonts.google.com/>
 - https://developers.google.com/fonts/docs/getting_started

```
<style type="text/css">
  @import url(css/s03.css);
</style>
```

Selettori



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

```
div>span { ... }  
div span { ... }  
#myId + p { ... }
```

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

- Selezione degli elementi nella pagina a cui applicare la regola:
 - Nome del tag
 - Classe, attributo class [punto]
 - Identificatore, attributo id [cancelletto]
 - Attributo (presenza, valore specifico)
 - Pseudo classe [:] (hover, checked, nth-child(), ...), anchor → a:link, a:visited
 - Pseudo elemento [::] (before, after, selection, first-letter, ...)
 - Discendenza diretta [>]
 - Discendenza generica [spazio]
 - Elemento successivo [+], tutti quelli specificati allo stesso livello [~]
- Più selettori possono essere associati a una regola
 - L'operatore **virgola** viene interpretato come un **OR** inclusivo
 - La **concatenazione** semplice si interpreta come un **AND**

Cascading

Più regole per lo stesso selettore, vince l'ultima

```
p {  
  font-family: Arial;  
  font-size: 12px;  
}
```

```
p {  
  color: red;  
  font-size: 11px;  
}
```

```
p {  
  margin-left: 15px;  
  color: green;  
}
```

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

!important

Specificità: la priorità va al selettore più specifico

```
p {  
  color: green;  
}
```

```
.red {  
  color: red;  
}
```

```
<p class="red">Hello</p>
```

Ereditarietà: di (alcune) proprietà CSS

```
<p class="red">Hello <span>hello</span></p>
```

Uso di id e class


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

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

```
<div>  
  <p>  
    A few words with  
    <span id="special">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>
```

Funzioni

- `url()`
 - conversione esplicita di una stringa in URL: `url('https://.../star.gif')`
- `rgb()`
 - Notazione funzionale per definire colori [red, blue, green]
 - `rgb(255, 0, 0)`, `rgb(100%, 0%, 0%)`
- `attr()`
 - Valore di un attributo (supporto diffuso solo per content)
 - `<p target="hello">world</p>`
 - `[target]::before { content: attr(target) " "; }`
- ...



hello world

Proprietà

- Alcune tra le proprietà più usate in CSS:
 - **background**: sfondo di un elemento
 - **background-image**: url("img/myBackground.jpg"), **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, red, rgb(0, 0, 0), #FFFFFF
 - **font**: proprietà del carattere per il testo nell'elemento
 - **font-size** (18px), **font-family** (sans-serif), **font-style** (normal, italic), **font-weight** (bold, [100 .. 900])
 - **margin** e **padding**: spazio attorno all'elemento (esterno e interno ai bordi)
 - **line-height**: gestione dell'interlinea
 - **text-align** (left, center, right, justify): allineamento del testo
 - **text-transform**: (uppercase, capitalize), **text-decoration**: (none, underline, overline, line-through)
 - **width**, **height**: dimensioni dell'elemento, quando applicabili

Unità di misura

- **px**: pixel, assoluto. Definito come 1/96° di pollice
- approssimazione della lettera 'M'
 - **em**: relativa al padre
 - **rem**: relativa all'elemento root (HTML o BODY, a seconda del browser)
 - Non supportata dai browser più vecchi
- **%**: percentuale relativa al padre, se presente
- **vh**, **vw**: viewport height e width, in percentuale
 - Non supportata dai browser più vecchi
- ...

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 esterno all'elemento
 - **border**: bordo (invisibile per default)
 - **padding**: spazio interno attorno al contenuto
 - content: testo o elementi contenuti dell'elemento
- Le dimensioni (height e width) sono determinate da:
 - Tipo dell'elemento (block, inline)
 - Dimensioni del content

Liste

- Proprietà per stilare gli elementi LI
 - list-style-type
 - In UL: disc, circle, square, none, ...
 - In OL: decimal, lower-roman, lower-alpha, ...
 - list-style-position
 - inside, ...
 - list-style-image
 - Poco configurabile → background-image

La proprietà display

- Permette di modificare il “normal flow”
 - Per ogni elemento
 - Si esamina il content, gli si aggiunge il padding, border e margin
 - Se block: width = 100% del parent, height determinata dal content
 - Se inline: sulla riga corrente se c'è spazio, o sulla riga successiva
 - *Block flow direction*, basata sulla direzione di scrittura
- Per cambiare la modalità di un elemento, **display**:
 - block
 - inline
 - inline-block: inline con padding e margini su tutti lati (IMG)
 - flex
 - none

```
nav>ul>li {  
    display: inline-block;  
}
```

```
<nav>  
  <ul>  
    <li><a href="#">Home</a></li>
```

Flexbox

- Permette di disporre elementi in una dimensione
- **Flex container**, proprietà di visualizzazione flex:
 - **display: flex** → determina un flex container
 - flex-direction → direzione di scorrimento nel container (default: row)
 - flex-wrap → box su riga singola o più righe (default: nowrap)
 - justify-content → spazi tra e attorno gli elementi (default: normal)
 - align-items → allineamento lungo l'asse principale (orizzontale)
 - Default: stretch, per centrare gli item nel container → center
- **Flex items**, elementi inseriti in un flex container
 - flex sui box associati per determinare la loro dimensione minima (**flex: 33%**)

Grid

- Permette di disporre elementi su due dimensioni
- Se sono inseriti in un `grid container`, gli elementi usano il *layout grid* definito dalla proprietà
 - `display: grid` → container grid con una sola colonna
 - `grid-template-columns`: 1fr 2fr 1fr → tre colonne (dimensioni in frazioni)
 - Ripetizione di valori via funzione CSS: `repeat(3, 1fr)`;
 - `gap`: 20px → distanza tra elementi
 - `row-gap`: 10px;
 - `column-gap`: 10px;


float

- display: **flow-root**
 - Nel container, detto *block formatting context* (BFC)
 - Elemento float e altri (con eventuale clear)
- **float**: left | right → elemento rimosso dal normal flow
 - Posizionato a sinistra | destra nel suo container
 - Il resto del content che segue viene posizionato al suo fianco
- **clear**
 - Rimuove il float dall'elemento corrente
 - Va specificato a quale float mi riferisco (left, right, both)

Media query

- Regole CSS valide solo in certe condizioni
- Breakpoint: dimensione per cambio regola
 - 600px tra smartphone e tablet?
 - 1200px tra standard desktop e extra?
- Esempio, per piccoli schermi:
 - `@media screen and(max-width: 600px) { /* ... */ }`

Tools

- Preprocessori CSS
 - Less
 - SCSS / SASS 
 - Stylus
 - ...
- Minify
 - Riduce la dimensione del file
 - Mantiene il comportamento
 - Warning, errori
- <https://sass-lang.com/install>
- sass input:output
 - opzione watch
- Variabili
 - Ex: \$dark: darkblue
- Nesting
 - Simile alla struttura HTML
- Moduli, mixin, gerarchie
- Operatori
 - +, -, *, /, ...