CSS
Fondamenti

dott. ing. Massimiliano Kraus

Intro

Il linguaggio HTML ci permette di scambiare informazioni tra calcolatori, in modo dichiarativo, e con una ricca semantica (vedi relativa dispensa).

Si pongono però tre problemi, correlati:

Design

Una pagina web, anche solo mediamente complessa, diventa difficile da leggere e da navigare usando lo stile di default.

Estetica

Una pagina web "fatta bene", "bella" piace di più rispetto ad una pagina "povera", sproporzionata, non curata.

Marketing

Una pagina con design ed estetica migliori (layout accattivante, grafica semplice ma moderna, etc.) attira più lettori, e quindi più clienti.
Questo ha impatti enormi per le aziende che hanno il proprio business online.

Il CSS

Questi aspetti hanno portato fin da subito alla creazione di diversi linguaggi per poter scrivere i cosiddetti «fogli di stile», ovvero delle istruzioni per gestire la grafica di un documento HTML.

In particolare, il linguaggio che si è rivelato vincente nella seconda metà degli anni '90 è stato il CSS, il quale è diventato di fatto lo standard mondiale per la gestione della grafica web.

Anche il CSS, come l'HTML, è un linguaggio dichiarativo. Viene lasciato ad ogni browser l'implementazione effettiva degli stili; il CSS indica solo qual è il risultato voluto, in termini di forme, colori, dimensioni, disposizione degli elementi.

La sintassi (1)

Declaration

Una coppia proprietà-valore, nella forma: `proprietà: valore;`

La proprietà si riferisce alle proprietà degli elementi HTML che si vogliono modificare.

Ad esempio: `background-color: yellow;`

modifica il colore di sfondo degli elementi HTML selezionati, rendendolo giallo.

Declaration block

È una lista di dichiarazioni racchiuse da parentesi grafe. Es:

```
{
    background-color: yellow;
    font-size: 24px;
    display: inline-block;
    color: #AE4;
}
```

La sintassi (2)

Selector

Un blocco è preceduto da un selettore, che individua quali sono gli elementi HTML di cui modificare lo stile. Es: `body`

Ruleset

È l'insieme di un selettore ed il suo blocco. Es:

```
body {
    background-color: yellow;
    font-size: 24px;
    display: inline-block;
    color: #AE4;
}
```

Inserimento del CSS (1)

Il codice CSS può essere "agganciato" alla pagina web in tre modi.

Inline

Sfruttando l'attributo `style` presente in ogni elemento HTML (nel qual caso non va messo il selettore):

```
<h1 style="text-align:center;font-weight:bold;">Titolo</h1>
```

Nel tag `<style>`

Nel tag `<head>` è possibile inserire un tag `<style>` che racchiuda i ruleset utilizzati in tutta la pagina.

```
<style>
    p {
        font-family: "Menlo";
        margin: 20px;
    }

    .container {
        width: 1000px;
        max-height: 500px;
    }
</style>
```

Inserimento del CSS (2)

In un foglio esterno

Un foglio esterno contiene lo stesso CSS che conterrebbe il tag `<style>` dentro la pagina.

Per importare un foglio esterno va usato il tag `<link>`, sempre dentro lo `<head>` della pagina.

```
<link rel="stylesheet" href="~/css/style5.css" />
```

L'attributo `rel` è necessario per dichiarare la relazione tra la pagina web e il documento esterno importato.

Inserimento del CSS (3)

Quale modalità è meglio utilizzare?

Lo stile **inline** costringe a riscrivere lo stile per ogni elemento HTML, il che porta ad una duplicazione di codice enorme anche su pagine semplici.
Il tag **`<style>`** evita la duplicazione di codice dentro la singola pagina, perché ogni ruleset viene applicato a tutti gli elementi HTML individuati dal suo selettore. Però nel caso di un sito web multi-pagina, l'intero contenuto del tag **`<style>`** va ricopiato in ogni file HTML.
Un foglio esterno invece può essere importato in quanti file HTML si vuole, senza duplicazione alcuna.

In genere quindi si raccolgono tutti gli stili comuni del proprio sito web dentro dei fogli di stile esterni; quando una pagina ha bisogno di ruleset particolari, ci sono due opzioni: o si crea un foglio di stile esterno in più (che solo quella pagina importa), oppure si inserisce lo stile particolare nel tag `<style>`.

Selettori base (1)

Ci sono diversi tipi di selettori.

Selettori per *tag*

Selezionano tutti e soli gli elementi HTML con un certo tag.

Il ruleset qui sotto ad esempio si applica a tutti e soli gli elementi `<span>`.

Un selettore per tag è indicato con il semplice nome del tag stesso.

```
span {
    padding-left: 5px;
}
```

Selettori base (2)

Selettori per *classe*

Ogni elemento HTML ha un attributo `class` il cui valore può essere un elenco di classi CSS, separate da spazio.

Un selettore per classe seleziona tutti e soli gli elementi HTML che hanno l'attributo class contenente quella classe.

```
.header {
    margin-bottom: 15px;
}
```

```
<header>
    a text
</header>
<div class="header row">
    content of the div
</div>
<h6 class="title text-center header">
    a sub-title
</h6>
```

In questo esempio solo il `<div>` e l'`<h6>` ricevono lo stile, perché nel loro attributo `class` compare in elenco la classe `header`. Il tag `<header>`, pur chiamandosi così, non ha la classe `header` nell'attributo `class`, quindi NON riceve lo stile.

Selettori base (3)

Selettori per *id*

Ogni elemento HTML ha un attributo `id` il cui valore è una stringa identificativa.

La regola vuole che in ogni foglio HTML ci sia un solo elemento con un certo `id`, ma nel caso ce ne fosse più di uno, lo stile si applica a tutti.

Un selettore CSS per `id` seleziona tutti e soli gli elementi HTML che hanno l'attributo `id` corrispondente. Il selettore si crea introducendo l'`id` con il simbolo `#`.

```
#header {
    margin-bottom: 15px;
}
```

```
<header>
    a text
</header>
<main id="header">
    content of the div
</main>
```

In questo esempio solo il `<main>` riceve lo stile, perché il suo attributo `id` corrisponde a quello indicato nel foglio di stile. Il tag `<header>`, pur chiamandosi così, non ha l'`id`, quindi NON riceve lo stile.

Selettori base (4)

Pseudo-classi

Ogni elemento HTML può trovarsi in uno stato particolare. Ad esempio, può avere il mouse sopra in un certo momento, oppure può essere il figlio n-esimo del suo contenitore. Tutti questi stati particolari sono definiti *pseudo-classi* e sono introdotti dal simbolo ":".

Di solito si usano in combinazione ad un altro selettore.

```
a:hover {
    color: red;
}
```

In questo esempio, quando si passa con il mouse sopra qualsiasi elemento `a`, il testo di quell'elemento diventa di colore rosso.

Quando si toglie il mouse da quell'elemento, il colore ritorna quello di prima.

Selettori base (5)

Selettori per *attributo*

Si possono selezionare tutti gli elementi che hanno un certo attributo con un valore ben preciso, oppure tutti gli elementi hanno un attributo generalmente valorizzato (indipendentemente dal valore). Anche questo selettore in genere si usa insieme ad un altro.

```
img[display="inline"] {
    color: red;
}
```

In questo esempio, vengono selezionati tutti gli elementi `img` che possiedono l'attributo `display` con valore `inline`.

```
img[display] {
    color: red;
}
```

In questo esempio, vengono selezionati tutti gli elementi `img` che possiedono l'attributo `display`, indipendentemente dal valore.

Selettori composti (1)

I selettori di base si possono comporre insieme in diversi modi.

Vediamo i più semplici e comuni.

Selettori composti (2)

Selettori multipli

Un declaration block può essere associato a più selettori contemporaneamente: basta porre tutti i selettori prima del block, separati da virgole.

In questo modo si evitano duplicazioni di codice:

```
header {
    color: blue;
}
.logo {
    color: blue;
}
#main-title {
    color: blue;
}
```

```
header, .logo, #main-title {
    color: blue;
}
```

In questo esempio, il colore blu è applicato a tutti e soli gli elementi che sono di tipo `<header>`, oppure hanno la classe `logo`, oppure hanno come id `main-title`.

Selettori composti (3)

Selettori contenuti generalmente in altri selettori

La forma "`selettore1 selettore2`" permette di selezionare tutti e soli gli elementi individuati dal `selettore2` che sono anche discendenti del `selettore1`.

```
<style>
    div .text-center { text-align: center; }
</style>
<div>
    <h3 class="text-center">a text</h3>
    <span>
        <p class="text-center">another text</p>
    </span>
</div>
<h6 class="text-center">the last text</h6>
```

In questo esempio, l'allineamento al centro è applicato solo all'`h3` (figlio diretto del `div`) e allo `span` (discendente più lontano). NON è applicato al tag `h6`, perché pur avendo la classe `text-center`, non è contenuto in un `div`.

Selettori composti (4)

Selettori contenuti direttamente in altri selettori

La forma "`selettore1 > selettore2`" permette di selezionare tutti e soli gli elementi individuati dal `selettore2` che sono anche discendenti diretti del `selettore1`.

```
<style>
    .row > div { background-color: orange; }
</style>
<footer class="row">
    <div>a text</div>
    <span>
        <div>another text</p>
    </span>
</div>
```

In questo esempio, il colore di sfondo arancione è applicato solo al primo `div` (figlio diretto dell'elemento con classe `row`) e non al secondo (che è un discendente più lontano).

Selettori composti (5)

Concatenazione

Si possono concatenare quanti selettori si vogliono.

```
header > div .section #myimg:hover {
    color: blue;
}
```

In questo esempio, viene mostrato in blu il contenuto di tutti gli elementi con id `myimg`, contenuti in qualsiasi elemento con classe `section` contenuto in un tag `<div>` figlio diretto di un tag `<header>`, ma solo quando il mouse passa sopra l'elemento con id `myimg`.

Le regole di Cascading (1)

CSS è l'acronimo di **Cascading Style Sheets**, che significa "stili applicati a cascata".

Il CSS viene applicato "a cascata" secondo molteplici punti di vista, qui spiegati.

Somma degli stili

La prima regola è : quando selettori diversi influenzano un elemento, gli stili si "sommano".

```
.class1 {
    color: red;
}
#id1 {
    background-color: blue;
}
```

Un elemento che abbia sia classe `class1` che id `id1`, vedrà cambiati sia il colore del testo che lo sfondo.

Le regole di Cascading (2)

Tipo di selettore

Hanno precedenza le dichiarazioni fatte *inline*.

Seguono i selettori per *id*, poi quelli per *classe* e infine quelli per *tag*.

```
.class1 { color: red; }
#id1 { color: blue; }
h1 { color: green; }
```

```
<h1>a text</h1>
<h1 class="class1">some text</h1>
<h1 class="class1" id="id1">some text</h1>
<h1 class="class1" id="id1" style="color: violet;">some text</h1>
```

I vari `h1` avranno nell'ordine il colore: verde, rosso, blu, viola.

Le regole di Cascading (2-bis)

Tipo di selettore con selettori composti

Nel caso di conflitto tra selettori composti, le regole sono le seguenti:

- vince il selettore con il maggior numero di *id*;
- a parità di *id*, vince il selettore col maggior numero di *classi*;
- a parità di *classi*, vince il selettore col maggior numero di *tag*;
- i selettori per *pseudo-classe* e per *attributo* contano come le *classi*

Gli esempi seguenti di selettori composti sono in ordine di precedenza:

```
#container1 #main-logo .row div h4

main .container #main-logo .row div h4

.container #main-logo div span h4

main section article div span h4
```

Le regole di Cascading (3)

Ordine

A parità di selettore, selettori posti dopo nel codice sovrascrivono le impostazioni dei selettori precedenti.

```
#id1 { color: green; }
.class1 { color: red; }
.class2 { color: blue; }
```

```
<h1 class="class1">some text</h1>
<h1 class="class1 class2">some text</h1>
<h1 class="class2 class1">some text</h1>
<h1 class="class2 class1" id="id1">some text</h1>
```

Il primo `h1` è rosso.

Il secondo e il terzo `h1` hanno entrambi colore blu, anche se hanno l'ordine delle classi invertito, perché nel CSS la classe `class2` è dichiarata dopo e quindi la sua dichiarazione sul colore sovrascrive quella di `class1`.

Il quarto `h1` ha colore verde: anche se `class2` viene dopo `id1`, `id1` è di tipo diverso (tipo *id* invece di tipo *classe*) e quindi si applica la regola precedente sul tipo di selettore, secondo la quale i selettori per *id* hanno precedenza su quelli per *classe*.

Le regole di Cascading (4)

Pseudo-classi

Le *pseudo-classi*, se applicate a selettori di tipo *tag* o *classe*, sovrascrivono le impostazioni di tutti i selettori di tipo *tag* e *classe*, ma non quelli di tipo *id* o *inline*.

Se è presente un tag `h1` con classe `title`, un selettore `h1:hover` ha precedenza su un selettore `h1` o su un selettore `.title`, ma se l'`h1` ha anche un selettore per *id* che setta alcune proprietà, oppure se c'è dello stile *inline*, la pseudo-classe non può sovrascriverle.

Una *pseudo-classe* applicata ad un selettore di tipo *id* sovrascrive le impostazioni degli altri selettori per *id* applicati allo stesso id, anche se questi ultimi sono scritti dopo.

Le regole di Cascading (5)

Dal contenitore al contenuto

Un contenitore in molti casi "riversa" a cascata il proprio stile sugli elementi contenuti, sovrascrivendo quelli del suo contenitore.

```
<div style="color: red;">
    <div style="color: green">
        <p>some text</p>
    </div>
    <p>some other text</p>
</div>
```

Il secondo `p` ha colore rosso, ma il primo ha colore verde perché l'impostazione del `div` più esterno è sovrascritta da quella del `div` interno.

Le regole di Cascading (6)

File esterni

Le regole enunciate finora valgono per tutti gli stili associati ad una pagina HTML, indipendentemente se siano posti in un tag `<style>` o in un foglio esterno.

Se sono importati due fogli esterni di stile, il `<link>` o lo `<style>` inserito per ultimo sovrascrive quelli precedenti, secondo le regole mostrate.

Quindi, quando caso si utilizzino delle librerie CSS pronte (come Bootstrap), è opportuno prima importare le librerie, e poi i propri fogli di stile, in modo da poter sovrascrivere la libreria nel caso in cui si vogliano personalizzare alcuni aspetti dello stile.

Le regole di Cascading (7)

Il modificatore `!important`

Ogni dichiarazione CSS può essere marcata con il modificatore **`!important`** prima del ";", così: `p { color: red !important; }`

Le dichiarazioni marcate in questo modo hanno precedenza su tutte le altre.

In questo caso, un certo paragrafo `p` potrebbe avere associate dichiarazioni CSS per *classe, id* o *inline*, che normalmente avrebbero precedenza sul selettore indicato sopra che è di tipo *tag…* ma il colore rosso indicato avrà precedenza su tutti, in quanto marcato con `!important`.

Se due selettori hanno entrambi la stessa proprietà marcata con `!important`, si applicano le regole viste in precedenza (quindi un selettore per *id* "vince" su un selettore per *classe*, ecc).

Se ad esempio oltre al ruleset sopra fosse presente anche il seguente, quest'ultimo vincerebbe: `#my-paragraph { color: green !important; }`

(Per chi sa giocare a carte, usare `!important` ha lo stesso funzionamento di una briscola).

Le regole di Cascading (8)

L'origine

Ogni browser ha dei fogli di stile usati di default, chiamati **User Agent Styles**.

Questi stili sono applicati se il CSS non indica esplicitamente altrimenti.

Ecco spiegato perché un tag `h1` viene mostrato più grande e in grassetto, un elenco non ordinato viene visualizzato con dei dischetti, gli hyperlink sono blu e sottolineati, ecc ecc.

L'insieme delle dichiarazioni CSS espresse esplicitamente (che sono scritte dallo sviluppatore del sito web) vengono chiamate **Author Styles**, e hanno sempre precedenza sugli User Agent Styles (che hanno origine dal browser).

Ereditarietà

Nel CSS non esiste il concetto di "ereditarietà" come lo conosciamo dai linguaggi di programmazione, nel senso che non è possibile far derivare lo stile di un selettore dallo stile di un altro selettore.

Per applicare più stili ad un elemento, la soluzione è agganciargli più id/classi CSS.

In questo senso il CSS non lavora per ereditarietà ma per composizione.

Esiste il valore **`inherit`** per le dichiarazioni di stile, ma significa: "eredita il valore di questa dichiarazione *dal tuo contenitore*". Non è quindi un'ereditarietà in senso classico.

Oltre il CSS

Per meglio strutturare grafiche complesse di certe dimensioni, sono state create diverse «evoluzioni» del CSS che permettono migliore organizzazione degli stili e minore duplicazione.

In particolare, le più usate sono:

- **SASS** (Super Awesome Style Sheet)
- **LESS** (LEaner Style Sheet)

SASS e LESS non sono altro che delle estensioni del CSS, che aggiungono variabili, ereditarietà, annidamento, funzioni, ecc.
Il codice scritto in questi due linguaggi viene pre-processato e tradotto in CSS classico che i browser possono utilizzare per modificare la grafica della pagina.

Riferimenti

I seguenti siti web contengono informazioni di dettaglio sulle diverse declaration **CSS**, la loro applicabilità, i browser che le supportano, esempi ed esercizi:

https://www.w3schools.com/css/default.asp

https://developer.mozilla.org/en-US/docs/Web/CSS

Su **SASS**:

http://sass-lang.com/

Su **LESS**:

http://lesscss.org/