

|  |  |
| --- | --- |
| **Introduzione CSS** | |
| **Sintassi del CSS**  Il CSS consiste in una lista di statements, di cui si hanno due tipi:   * *at-rules* * *rule sets* |
| **at-rules**   * Regole introdotte attraverso una chiocciola e un identificatore * La dichiarazione si conclude con un punto e virgola o con un blocco delimitato da parentesi graffe che contiene proprietà della regola. |
| **rule sets**   * Consiste in un selettore (gli elementi a cui andrò ad applicare le regole presenti nel blocco di dichiarazione) seguito da un blocco (delimitato da parentesi graffe) di dichiarazione. * Se il selettore non viene riconosciuto il browser ignora la riga in cui si trova e il blocco di dichiarazione. * Una dichiarazione consiste nel definire una proprietà grafica associata al selettore: abbiamo una colonna col nome della proprietà, separata dal valore associato attraverso i due punti. * In un blocco di dichiarazioni le varie dichiarazioni sono separate da punti e virgola. |
| **Dichiarazioni**   * Una dichiarazione presenta la seguente struttura nome: value; dove il *nome* consiste nel nome della proprietà dichiarata. * Sono tollerati spazi bianchi attorno alle dichiarazioni * I browser ignorano dichiarazioni che presentano un nome e/o un valore sbagliato. |
| **Commenti**   * I commenti sono introdotti attraverso i seguenti simboli /\* COMMENTO \*/ * Possono essere introdotti in qualunque punto: non avranno influenza sul rendering. * Non sono possibili innesti. |

|  |
| --- |
| **Selettori** |
| * Il selettore si trova in una rule sets e permette di determinare a quali elementi del documento HTML si applicheranno certe proprietà[[1]](#footnote-1). * Il selettore può essere immaginato come una catena di uno o più selettori semplici separati da combinatori. * Il selettore semplice è seguito da zero o più attributi selettori, selettori di ID o pseudo-classi e può essere di due tipi:   + selettore di tipo;   + selettore universale. |
| **Sintassi dei selettori**   * Presente nelle diapositive di Marcelloni e in quelle di Tesconi riassunti delle possibili sintassi per i selettori.   + Marcelloni fornisce una lista completa con spiegazione (in inglese, va bè)   + Tesconi riporta la lista dei selettori più importanti, quelli che vale la pena mettersi nel capo. * Trovate queste diapositive qualche pagina più avanti nella dispensa. |
| **Combinatori**  **Esempi**: body > p { … }  .elemento + .elementosuccessivo { … }   * I combinatori sono simboli presenti nei selettori   + Lo spazio bianco   + La parentesi angolare di chiusura >   + Il simbolo di somma + |
| **Selettore di tipo**  **Esempi**: body { … } p { … }   * Il selettore di tipo coincide col nome di un elemento già definito dal linguaggio HTML. * Non è preceduto da alcun simbolo. |
| **Selettore universale**  \* { … }   * Il selettore universale, l'asterisco, permette di applicare certe proprietà a un qualunque elemento HTML. |
| **Raggruppamento di selettori**   * Se più selettori presentano le stesse dichiarazioni (cioè le stesse proprietà) allora possiamo unire il tutto in un unico *rule set* avente per selettore la lista dei selettori separati da virgola. * **Esempio**:  h1 { font-family: sans-serif }  h2 { font-family: sans-serif }  h3 { font-family: sans-serif }   h1, h2, h3 {  font-family: sans-serif; } |
| **Selettore di classe**   * Se ci si limitasse a selettori di tipo e selettori universali si avrebbe l’obbligo di stabilire proprietà per tutti i paragrafi possibili, per tutte le sezioni possibili, per ogni elemento possibile… Questa cosa può essere superata ricorrendo alle classi! * Possiamo fissare a livello di CSS uno stile di presentazione valido per tutti gli elementi associati a una classe. Per esempio:     Tutti i nomi di classi sono preceduti da un punto.   * Possiamo indicare più classi all'interno del selettore (cioè stabilire il manifestarsi di uno stile grafico se un elemento è associato a più classi in contemporanea). Vedere l’esempio.   **Esempio**:  p.green.bold { color: green; font-weight: bold } <p class="green serif bold"> *(OK, il paragraph è associato sia alla classe green che alla classe bold)*  <p class="serif bold"> *(NO, il paragraph è associato solo alla classe bold)* |
| **Selettore di ID**   * Il selettore ID fa riferimento all'attributo id visto negli elementi HTML * I selettori ID sono introdotti da un cancelletto #. * Esempio: <style type=”text/css”> #black { color: black; } </style> … <p class=”red” id=”black”>This paragraph is black</p> |
| **Pseudo-elementi**   * Gli pseudo-elementi creano delle astrazioni sull'albero DOM che vanno oltre ciò che possiamo specificare attraverso il linguaggio del documento. * Per esempio, non abbiamo meccanismi per raggiungere la prima lettera di una prima riga all'interno del contenuto di un elemento. (la prima lettera della prima riga non è contenuta in un elemento HTML) * **Esempi di pseudo-elementi**:   + :first-line (ponendo come selettore p:first-line possiamo applicare delle proprietà alla prima linea di ogni paragrafo) p:first-line {  … }   + :first-letter (ponendo come selettore span:first-letter possiamo applicare delle proprietà alla prima lettera presente in uno span) span:first-letter {  … }   + :before e :after, permettono di aggiungere qualcosa prima o dopo il contenuto di un certo elemento (attraverso la dichiarazione di stile content: ) p:after {  content: “ciao”; } |
| **Pseudo-classi**   * Le pseudo-classi permettono di classificare elementi con caratteristiche che vanno oltre il loro nome, gli attributi o il contenuto. * **Differenza rispetto agli pseudo-elementi**: gli pseudo-elementi fanno riferimento a cose non associabili ad elementi HTML, le pseudo-classi permettono di selezionare particolari elementi HTML (per esempio con div:first-child associamo proprietà stilistiche al primo elemento contenuto in un elemento div). * **Esempi di pseudo-classi**:   + :first-child, primo figlio all'interno di un elemento.   + :link, link non ancora visitati   + :visited, link che sono già stati visitati   + :hover, quando si passa col cursore sopra un elemento   + :active, clicchiamo col cursore sopra l'elemento ma non abbiamo ancora sollevato il dito dal tasto del mouse   + :focus, abbiamo cliccato a tutti gli effetti l'elemento (click e rilascio)   + :target, stile di presentazione riferito al target di una specifica anchor (cioè clicco un link che rimanda a una sezione del documento, questa sezione cambia stile) |

|  |
| --- |
| **Come si associa uno stile di rappresentazione a un certo elemento?** |
| * I due meccanismi che consentono di identificare in modo univoco lo stile di rappresentazione  sono il *cascading* e *l’ereditarietà*. * I meccanismi sono pensati in modo tale da risolvere eventuali situazioni di conflitto. |
| **Cascading (meccanismo che da il nome al CSS)**   * Le dichiarazioni di stile vanno a cascata su un elemento da più origini. * Esiste un meccanismo, abbastanza complicato, dove ogni dichiarazione viene pesata in base a una serie di fattori:   + l'ordine con cui le fonti sono stati introdotte   + la sua importanza.   + l'origine   + la specificità   la cosa con peso più alto sarà adottata dal browser.   1. Per ogni elemento il browser individua tutte le dichiarazioni associate a uno specifico elemento **analizzando tre sorgenti**:    * il browser    * l'autore    * i fogli di stile degli utenti (un set di valori che alcuni browser permettono di personalizzare)   Le dichiarazioni **sono adottate** se   * + il selettore associato soddisfa l'elemento, e…   + il mezzo con cui stiamo fruendo del documento è presente nella lista dei media su tutte le regole media o su tutti i link attraverso cui accediamo agli stylesheet.  1. Se ci sono più dichiarazioni applicabili ordino le dichiarazioni in base alla loro **importanza** e **origine**.   Classifichiamo le dichiarazioni:   * + Rispetto all’importanza:     - Dichiarazioni importanti, introdotte nella forma property: value!important;     - Dichiarazioni normali, dove !important non è presente   + Rispetto all’origine:     - Dichiarazioni del browser: stylesheet di default del browser (valori di default di tutte le proprietà). Proprietà visibili facendo *Ispeziona elemento*. Vediamo un esempio:     Dichiarazioni del browser relative al body.  La proprietà margin è sovrascritta da altre proprietà indicate dal programmatore.   * + - Dichiarazioni dell’utente: dichiarazioni che gli utenti possono inserire in alcuni browser (stylesheet personalizzati, era una cosa che andava molto dieci anni fa…)     - Dichiarazioni dell’autore: ciò che scriviamo NOI programmatori nel foglio CSS.   Le dichiarazioni sono ordinate in modo crescente secondo questa lista   * 1. Dichiarazioni di transizione   2. Dichiarazioni importanti del browser   3. Dichiarazioni importanti dell'utente   Dichiarazioni !important   * 1. Dichiarazioni override importanti   2. Dichiarazioni importanti dell'autore   3. Dichiarazioni di animazione   4. Dichiarazioni di override normali   5. ***Dichiarazioni normali dell'autore***   Dichiarazioni normali   * 1. *Dichiarazioni normali dell'utente*   2. *Dichiarazioni normali del browser.*   Le ultime tre sono quelle che ci devono accendere la lampadina. In fondo alla classifica troviamo le dichiarazioni normali del browser, cioè lo stylesheet di default offerto dal browser: noi programmatori, quando scriviamo CSS, andiamo a sovrascrivere proprio quelle proprietà. Ciò che scriviamo noi ha precedenza rispetto allo stylesheet di default.   1. Le dichiarazioni che presentano lo stesso livello di importanza e di origine sono ordinate analizzando la **specificità** del selettore. La specificità si calcola a partire da quattro valori separati da virgola: .  I valori in sono i più importanti, quelli in i meno importanti.    * se la dichiarazione si ottiene da un attributo style e non da una regola con selettore.    * consiste nel numero di attributi ID nel selettore    * consiste nel numero di altri attributi e pseudo classi nel selettore.    * consiste nel numero di nomi elemento e pseudo elementi nel selettore.   Vediamo un esempio di graduatoria (dalla minore alla maggiore specificità):       * + - Si guarda il valore di a per trovare le dichiarazioni più specifiche.     - A parità di a si guarda il valore di b     - A parità di b si guarda il valore di c     - In caso di parità di c si guarda il valore di d  1. A questo punto se le dichiarazioni hanno lo stesso livello di importanza, origine e specificità, si ordinano in base all’inserimento nel codice. Vince l'ultima dichiarazione introdotta. |
| **Esempio di cascading (specificità)**  Il primo paragraph ha id redP e class bluStyle, oltre ad avere l’attributo style con ulteriori dichiarazioni di CSS. Ecco la graduatoria delle dichiarazioni (dalla meno importante a quella con priorità più alta):   1. Proprietà della classe bluStyle (colore blu del font) 2. Proprietà dell’id redP (colore rosso del font) 3. Proprietà indicate dall’attributo style (colore verde del font).   **Risultato**: **Example 1** di colore verde.  <!DOCTYPE HTML>  <html>  <head>  <style type="text/css">  #redP { color: red }  p.bluStyle {color:blue}  </style>  <title>Resolution Process</title>  </head>  <body>  <p id="redP" class="bluStyle" style="color:green">  Example 1 </p>  <p class="bluStyle"> Example 2 </p>  </body>  </html> |

|  |  |
| --- | --- |
| **Ulteriore esempio di Cascading (specificità)**  **Primo paragraph**:   * Abbiamo un selettore di tipo (p) con cui definiamo padding e bordo  (fin qua nessun conflitto). * Il colore del testo è quello definito dalla dichiarazione del selettore #wrap, che vince sul selettore di tipo body (i selettori di tipo hanno minore importanza rispetto ai selettori di ID). * Anche la dimensione del testo è indicata dalla dichiarazione associata al selettore #wrap, che vince sul selettore di tipo div (stesse motivazioni).   <!DOCTYPE HTML>  <html>  <head>  <style type="text/css">  body {  color: #000;  background-color: #fff;  }  #wrap {  font-size: 2em;  color: #333;  }  div { font-size: 1em; }  em { color: #666; }  p.item {  **Secondo paragraph**:   * Le proprietà del paragraph sono le stesse del primo, con le dovute differenze relative alle classe item:   + Lo stile del bordo (*dashed*, nessun conflitto)   + Il colore di sfondo (niente conflitti, vedere le proprietà più avanti)   + Il colore del font: in questo caso il selettore p.item vince sul selettore #wrap (immaginatevi di fare la graduatoria a pagina prima con a,b,c,d)   + Il font-size è lo stesso del primo paragraph. * L’elemento em presenta il colore indicato assieme al selettore di tipo em.   color: #fff;  background-color: #ccc;  border-style: dashed;  }  p {  border: 1px solid black;  padding: 0.5em;  }  </style>  </head>  <body>  <div id="wrap">  <p>Normal Paragraph</p>  <p class="item">  This is the <em>cascade</em> in <a href="#">action</a>  </p>  </div>  </body>  </html>    L’elemento HTML con id wrap contiene due paragraph. Il secondo paragraph appartiene alla classe item. Sempre all’interno del secondo paragraph è utilizzato l’elemento em.  Anteprima del codice | |
| **Ereditarietà**   * Con ereditarietà intendiamo il processo attraverso cui un certo elemento figlio eredita proprietà da un elemento padre: stabiliamo per un elemento proprietà non associate ad esso direttamente. * La cosa è evidente dall'albero DOM, con cui stabiliamo una gerarchia * Se stabiliamo il colore del font nel body, per esempio, questo colore sarà applicato a tutti gli elementi contenuti in body.   Vedremo, studiando le proprietà del CSS, che in certi casi (con certi oggetti e/o certe proprietà) si ha l’ereditarietà, in altri casi no. | |
| **Unità di misura** | | |
| **Unità relative** | |
| em | Dimensione del font rilevante (precisamente quello usato di default nel browser, porre 10em significa porre una dimensione dieci volte quella del font rilevante) |
| ex | Altezza del font rilevante: porre 10ex significa porre come dimensione dieci volte quella dell’altezza del font rilevante) |
| **Unità assolute** | |
| in | Inches, pollici (1in uguale a 2.54cm) |
| cm | Centimetri |
| mm | Millimetri |
| pt | Points (1pt equivale a 1/72esimo di 1inch) |
| pc | Pica (1pc equivale a 12pt) |
| px | Pixels (1px equivale a 1/96esimo di 1inch). Il pixel fisico è diverso da questo pixel: le dimensioni sono diverse (potrei utilizzare più pixel fisici, per esempio in una stampante laser, per ottenere il pixel grandezza) |

1. E se avessi conflitti? Generalmente hanno la precedenza le regole associate al selettore più specifico (per esempio ID vs CLASS) [↑](#footnote-ref-1)