CORSO HTML5 CSS3

DOCENTE: NICOLA CIACO

I Fogli di Stile

Per stile, si intende un **gruppo di istruzioni di formattazione**, archiviato con un certo nome, che consente di assegnare **una struttura predefinita** ad un documento e di **apportare modifiche all'aspetto del documento**.

Da un punto di vista concettuale, i fogli di stile sono dei modelli, sul tipo di quelli usati in molti elaboratori testo.
Come questi, essi consentono di attribuire proprietà di formattazione in aggiunta o a modifica di quelle predefinite e di semplificare le operazioni di manutenzione dei documenti.

I fogli di stile possono essere inseriti nel documento HTML o, più vantaggiosamente, essere definiti in file esterni (**nome_stile.css**).

In questo modo, una volta assegnata l'impostazione generale di una tipologia di documenti, sarà sufficiente apportare la modifica della proprietà al foglio di stile perché si ripercuota su tutti i documenti associati.

Come successo per l'HTML, arrivato oggi alla sua quinta *release*, anche i fogli di stile sono stati implementati negli anni. Oggi la versione più aggiornata è definita con il nome di CSS3.

I Fogli di Stile

In dettaglio, i fogli di stile consentono di:

- impostare la dimensione dei caratteri, i margini, i rientri, i colori di sfondo del testo, ...
- modificare l'aspetto di uno o più documenti HTML, o, magari, di un intero sito, senza dover esaminare e modificare decine e decine di righe di codice
- limitare il numero di marcatori all'interno dei documenti HTML
- impostare più attributi per ciascun marcatore e definire così delle classi di testo

Invece di applicare una formattazione a *ciascun paragrafo,* è possibile creare uno stile che applichi contemporaneamente i formati desiderati. Gli stili garantiscono inoltre una formattazione uniforme nel documento.

Il termine **Cascading** - a "cascata" (CSS è l'acronimo di Cascading Style Sheets) - richiama una delle caratteristiche principali di questa tecnologia: è possibile incorporare nel documento differenti istruzioni di stile, ognuno dei quali, in base a precise regole di gerarchia, viene ereditato dagli altri elementi.

I Fogli di Stile

L'ordine con cui vengono interpretati gli stili è il seguente:

- **Stili di default**
  Sono le regole di base che vengono applicate automaticamente dal Browser

- **Stili *in linea***
  Sono regole CSS scritte direttamente nei TAG e che nascono e muoiono con l'apertura e la chiusura di quel TAG.

- **Stili incorporati**
  Sono regole di stile raccolte solitamente nell'<head> del documento.
  Queste regole valgono esclusivamente per il documento in cui sono presenti.

- **Stili collegati**
  Sono e veri e propri fogli di stile, ovvero dei file .css che raccolgono tutte le informazioni estetiche e che vengono collegati a tutte le pagine del sito web.
  In questo modo TUTTO IL SITO WEB raccoglie le informazioni del foglio di stile collegato.

Applicare i fogli di stile

Gli stili in linea
Il concetto di stile in linea è molto vicino alle regole base dell'HTML: gli stili agiscono sui singoli tag che li contengono.
I fogli di stile in linea, quindi, agiscono su singole istruzioni senza influenzare in alcun modo la totalità della pagina.
Per inserire singole istruzioni di stile in un documento HTML è necessario usare l'attributo style="..." all'interno di ciascun TAG.

```
<body>
    <p style="text-align:center"> testo centrato </p>
</body>
```

In questo esempio il contenuto testuale del paragrafo verrà centrato rispetto al <p>

Concettualmente questo sistema di utilizzo degli stili non è scorretto, ma non sfrutta appieno le potenzialità offerte dal CSS né, tantomeno, risponde alla vera natura del CSS: uniformare lo stile di un certo numero di documenti o almeno di tutti i TAG di un certo tipo di un intero documento.

E' consigliabile limitare (se proprio non se ne possa fare a meno!) l'uso dello stile in linea per piccoli ritocchi e non inserirlo sistematicamente sulle singole istruzioni per non appesantire il codice.

E' bene tener presente che lo stile in linea limita tantissimo il Responsive Web Design.

Applicare i fogli di stile

Gli stili incorporati
La caratteristica fondamentale degli stili incorporati, è che essi vengono assegnati all'intero documento che li contiene e non ai singoli TAG HTML.

Rispetto alla normale sintassi HTML possiamo notare alcune peculiarità:
- gli attributi sono inseriti tra parentesi graffe {}
- al posto del segno = vengono usati i due punti per separare un attributo css dal suo valore
- gli argomenti consecutivi sono separati dal punto e virgola ;
- gli attributi composti da più termini sono separati da un trattino –

```
<head>
    <style>
        h1{ color:red ; text-align:center}
    </style>
</head>
<body>
    <h1>Questo titolo è ROSSO e CENTRATO</h1>
</body>
```

Per incorporare un foglio di stile in un documento HTML è necessario:
- inserire un marcatore <style>...</style> all'interno della sezione <head>...</head>;
- all'interno del marcatore <style>...</style> inserire le definizioni dei vari stili.
- Le informazioni contenute nel marcatore <style>...</style> vengono interpretate ma non visualizzate/stampate dal browser, per cui i browser che non prevedono il supporto dei fogli di stile ignoreranno semplicemente le istruzioni contenute all'interno del marcatore <style>…</style>.

Applicare i fogli di stile

Gli stili esterni
Per collegare un foglio di stile "esterno" a un documento HTML è necessario:

- creare un file contenente le definizioni dei vari stili;
- salvarlo con estensione .CSS;
- inserire un'istruzione di tipo <link rel="stylesheet" href="URL_foglio_di_stile.css" type="text/css"> nella sezione <head>...</head>

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

<body>
    …contenuto della pagina
</body>
```

- L'attributo href del marcatore <link> permette di specificare l'URL del foglio di stile da allegare/collegare al documento.
- L'attributo rel="stylesheet" fa sì che il browser interpreti la pagina collegata come style sheet e non come altro genere di include.

Applicare i fogli di stile

Gli stili importati
Per importare un foglio di stile in un documento HTML è necessario usare l'istruzione @import url(URL) all'interno del marcatore <style>...</style>
dove URL è il percorso in cui si trova il foglio di stile da importare.

```
<head>
<style>@import url(URL_foglio_di_stile.css)</style>
</head>
```

- L'attributo href del marcatore <link> permette di specificare l'URL del foglio di stile da allegare/collegare al documento.
- L'attributo rel="stylesheet" fa sì che il browser interpreti la pagina collegata come style sheet e non come altro genere di include.

Può accadere che un foglio importato abbia al suo interno un'istruzione @import.

In questo secondo caso si ottiene una serie di "strati" di stile la cui priorità si risolve in base all'ordine di dichiarazione: il CSS importato per ultimo sovrascrive quelli importati precedentemente.

Si possono importare più fogli di stile usando l'istruzione @import; ad esempio si possono inserire le regole di stile per i vari tag (immagini, sfondi, testo, ecc) in file di stile diversi ed importare l'uno oppure l'altro a seconda delle esigenze della pagina HTML

Sintassi di base

Una regola è una istruzione che riguarda l'aspetto stilistico di uno o più elementi.
Un foglio di stile è composto da una o più regole da applicare ad un documento HTML.

La regola è composta di due parti principali:

- selettore: la parte che precede la parentesi graffa aperta {
- dichiarazione: la parte all'interno delle due parentesi graffe{}

L'esempio che segue è un classico esempio di "**type selector**".
Ogni elemento HTML può essere usato come Type selector.

```
body{
        background-color:black;
}

h1{
        color:red;
        text-align:center
}
```

- **proprietà**: la parte prima dei due punti
- **valore**: la parte dopo i due punti

La proprietà è la qualità, o caratteristica, che un TAG possiede.
Il valore è una precisa specificazione della proprietà.
Nell'esempio accanto la proprietà background-color dell'elemento body è nera ma sarebbe potuta essere verde, gialla, ecc.

Sintassi di base

Uno degli scopi da raggiungere nel momento in cui si realizza un foglio di stile è la sinteticità delle istruzioni: riducendo la dimensione del foglio di stile la pagina HTML verrà caricata più velocemente.
Il CSS include diversi meccanismi per sintetizzare gli stili raggruppando selettori e dichiarazioni.

```
h1{
    color:red
}

h2{
    color:red
}

h3{
    color:red
}
```

Questo esempio di codice, in cui abbiamo un h1, un h2 e un h3 tutti con la proprietà color che serve a colorare il testo di rosso, può essere scritta in modo più sintetico, ottenendo lo stesso risultato:

```
h1, h2, h3{
    color:red
}
```

I selettori

Una regola CSS comunica con i TAG HTML **grazie all'uso di selettori.**
Grazie ai selettori è possibile, quindi, associare una qualsiasi regola CSS a un TAG (o a più TAG) presenti nel documento HTML.

Esistono vari tipi di selettori: è possibile comunicare al documento HTML le regole CSS in vari modi.

Il primo selettore che esamineremo è detto **selettore universale** e serve a selezionare tutti gli elementi di un documento.
Si esprime con il carattere * (asterisco).

```
*{ color: red }
```

La regola color: red scritta con un selettore *, indica a TUTTI I TAG HTML presenti nel documento di colorare di rosso l'eventuale testo in essi contenuto.

Il secondo tipo di selettore che esamineremo è detto **selettore di tipo** (o selettore di elementi).
È costituito dal nome di uno specifico elemento HTML (<h1>, <p>, <a>, <body>, …).
Serve a **selezionare tutti gli elementi di quel tipo presenti in un documento**.

```
p{ font-size: 14px; }
```

La regola font-size: 14px scritta con un selettore p, indica a TUTTI I TAG <p> presenti nel documento di settare l'eventuale testo in essi contenuto a una dimensione di 14 pixel.

I selettori

I selettori più utilizzati in assoluto sono senza dubbio i selettori **class** e **id**.

La definizione di classi consente di assegnare un nome a una o più proprietà rendendo possibile individuare univocamente determinate porzioni di testo e assegnare loro particolari stili.

Il funzionamento dell'attributo id è molto simile a quello dell'attributo class con un'importante differenza: **il valore di un attributo id deve essere univoco per l'intero documento**.

Ogni TAG presente in un documento HTML può avere un attributo id, ma nessun TAG può avere un id uguale a un altro TAG.

I valori assegnati a class e id sono testuali e del tutto arbitrari.
Sarà nostra discrezione, quindi, organizzare i nomi di questi due selettori.

Ogni elemento all'interno del TAG <HTML> può avere un attributo class; fondamentalmente si possono classificare elementi con l'attributo class, creando regole nel foglio di stile a cui l'elemento fa riferimento e il browser automaticamente applicherà queste regole al gruppo degli elementi di quella classe.

I selettori

Esempi di codice con i selettori class e id.

La sintassi per **class** è la seguente:
<style> .nome_class {elemento: proprietà;...elemento: proprietà} </style>

Il riferimento alle varie classi va poi eseguito secondo la sintassi:<etichetta class=" ">…</etichetta>
Il selettore di classe inizia con un carattere flag **.** (punto) che indica quale tipo di selettore segue.

```
<style>
    .miaclasse{color:red}
</style>

<body>
    <p class="miaclasse"> Questo testo è rosso </p>
</body>
```

L'attributo CLASS è una potente funzionalità del CSS.
Se ne raccomanda l'uso soprattutto nei casi in cui il blocco di codice CSS ad esso collegato può essere ripetuto più volte all'interno dello stesso documento.

I selettori

Ogni elemento, all'interno di un documento HTML, può avere un attributo **id** ma questi valori devono essere differenti gli uni dagli altri; conseguenza di questo è che l'attributo id è utile per **settare regole di stile su singoli elementi**.

La sintassi per **id** è la seguente:
<style> #nome_id {elemento: proprietà;...elemento: proprietà} </style>

Il riferimento alle varie classi va poi eseguito secondo la sintassi:<tag id=" ">…</tag>
Il selettore di ID inizia con un carattere flag **#** (cancelletto) che indica quale tipo di selettore segue.

```
<style>
    #mio_id{color:red}
</style>

<body>
    <p id="mio_id"> Questo testo è rosso </p>
</body>
```

Come si può notare, la sintassi di definizione è molto simile a quella dell'attributo class; il carattere flag però, nel caso di selettore di id, è costituito dal carattere cancelletto #, invece che punto .

Nota importante:
Non è possibile dare un valore che contenga uno spazio vuoto a un id o a una class.
Scrivere .mia classe{color: red} sarebbe un errore!)

I selettori

Sia per class che per id i valori saranno scelti arbitrariamente dai web designer.
Di fatto, spesso, i termini utilizzati sono funzionali anche a comprendere facilmente di quale blocco di codice HTML possa essere interessato.
Ecco un esempio di class e id con nomi semantici e con l'effetto *cascata*, tipico dei CSS

```
<style>
    #header{background-color: black; color: white}
     .testo-rosso{ color: red}
</style>

<body>
    <div id= "header">
        <h1 class= "testo-rosso">Titolo della pagina (rosso)</h1>
            <p>
            Il testo di questo paragrafo sarà bianco perché eredita
(cascading  style sheet) il colore dal TAG genitore <div id="header">
            </p>
            <p class="testo-rosso">
                    Il testo di questo paragrafo sarà rosso perché nel TAG è
richiamata una classe che ha come istruzione quella di colorare di
rosso il testo.
            </p>
    </div>
</body>
```

In questo esempio abbiamo realizzato un codice css legato al selettore id #header (che semanticamente ci fa pensare si tratti dell'elemento che costituirà la testata del sito web).
In questo blocco di codice è stato indicato che il colore di sfondo è nero e il colore del testo è bianco.

Tutti i TAG contenuti in questo TAG avranno (erediteranno a cascata) queste indicazioni.

Nel caso venisse, tramite una classe o un id, modificato un TAG contenuto nel , questa seconda istruzione prevarrebbe sulla prima (vincerà *sempre* l'istruzione più vicina al TAG!)

I selettori

Una categoria fondamentale di selettori CSS è rappresentata dai cosiddetti combinatori (detti anche selettori di relazione). Hanno la funzione di mettere in relazione elementi presenti all'interno dell'albero del documento.

Selettore	Simbolo
Selettore di discendenti	
Selettore di figli	>
Selettore di fratelli adiacenti	+
Selettore generale di fratelli	~

Il **selettore di discendenti** è sicuramente quello più utilizzato dei quattro. Seleziona un elemento che è discendente di un altro elemento (contenuto in un altro elemento a qualsiasi livello).
Per impostare la relazione di discendenza, è sufficiente separare l'elemento antenato dal discendente con uno spazio

```
<style>
    #contenitore p{ color: red }
</style>
```

Il selettore va letto per chiarezza **da destra a sinistra**. Il codice qui accanto serve ad assegnare lo stile (testo di colore rosso) **solo ai paragrafi (<p>)** contenuti nel , ovvero ai paragrafi discendenti del div con id contenitore

I selettori

Ecco un esempio più complesso di **selettore discendete:**

```
<style>
    ol ol {list-style: upper-alpha}
    ol ol ol { list-style: lower-alpha}
</style>
<body>
    <ol>
        <li>Primo livello
            <ol>
                <li>Secondo livello - Primo elemento</li>
                <li>Secondo livello - Secondo elemento
                    <ol>
                        <li>Terzo livello - Primo elemento</li>
                        <li>Terzo livello - Secondo elemento</li>
                    </ol>
                </li>
            </ol>
        <li>Secondo livello</li>
    </ol>
</body>
```

1. Primo livello
    a. Secondo livello - Primo elemento
    b. Secondo livello - Secondo elemento
        A. Terzo livello - Primo elemento
        B. Terzo livello - Secondo elemento
2. Secondo livello

Lo stile, in questo caso, influenza la numerazione degli elementi di secondo e terzo livello.

I selettori

Il **selettore di figli** (**>**) consente di selezionare un elemento che è figlio diretto dell'elemento padre.

Questo selettore è solo in apparenza simile al selettore di discendenti. La differenza sta nella relazione di discendenza tra gli elementi, che in questo caso deve essere di primo livello.

```
<style>
        #box > p {color: white}
</style>

<body>

        <div id="box">
            <p>Primo paragrafo</p>
            <div>
                <p>Secondo paragrafo</p>
            </div>
             <p>Terzo paragrafo</p>
        </div>

</body>
```

In questo esempio, dei 3 paragrafi, **solo** il **primo** e il **terzo** sono **figli diretti** del ** **.

Il secondo è invece figlio diretto di un elemento *anonimo*.

Tutti e tre, però, sono discendenti del con id #box

I selettori

Il **selettore di fratelli adiacenti** serve a scorrere in orizzontale l'albero del DOM assegnando le regole CSS agli elementi che si trovano allo stesso livello di un altro elemento.

La relazione si definisce collegando i due elementi con il segno **+**.

```
<style>
      h1 + h2 {color: white;}
</style>

<body>
   <div>
        <h1>1. Titolo principale h1.</h1>
        <h2>1.1 Primo sottotitolo h2.</h2>
        <p>...</p>
        <h2>1.2 Secondo sottotitolo h2 (non adiacente!).</h2>
         <p>...</p>
   </div>
</body>
```

In questo esempio la regola CSS collega tutti i TAG <h2> immediatamente adiacenti al TAG <h1>.

Per cui nel codice HTML succede che il TAG <h1> avrà il testo di colore bianco, così come il TAG <h2> adiacente ad esso.

Il secondo TAG <h2> (quello che è scritto qualche elemento dopo il TAG <h1>) non risulta essere adiacente, per cui ad esso non verrà applicata la regola CSS.

I selettori

L'ultimo selettore, caratterizzato dal simbolo **~** (tilde) è detto **selettore generale di fratelli** e potremmo dire che è una generalizzazione di quello visto in precedenza.
Infatti esso assegna uno stile a **tutti gli elementi che sono fratelli**, a prescindere dal fatto che siano o meno adiacenti.

Modifichiamo con questo selettore il codice scritto in precedenza:

```
<style>
      h1 ~ h2 {color: white;}
</style>

<body>
   <div>
        <h1>1. Titolo principale h1.</h1>
        <h2>1.1 Primo sottotitolo h2.</h2>
        <p>...</p>
        <h2>1.2 Secondo sottotitolo h2 (non adiacente!).</h2>
         <p>...</p>
   </div>
</body>
```

In questo esempio la regola CSS collega tutti i TAG <h2> fratelli al TAG <h1>.

Per cui nel codice HTML succede che il TAG <h1> avrà il testo di colore bianco, così come tutti i TAG <h2> adiacenti e non ad esso.

Per cui anche al secondo TAG <h2> (quello che è scritto qualche elemento dopo il TAG <h1>) verrà applicata la regola CSS.

I selettori

L'ultimo tipo di selettore, è definito **selettore di attributo**.
Questo selettore è riconoscibile perché oltre al nome dell'elemento presenta il nome dell'attributo, che va posto tra parentesi quadre (senza racchiuderlo tra virgolette).
Tra il nome dell'elemento e la definizione dell'attributo non va lasciato spazio.

Questo selettore di divide in **due sottotipi**.

Il **primo** agisce in base alla presenza di un attributo.

```
<style>
      a[title] {color: red;}
</style>
```

In questo esempio la regola CSS non andrà applicata a tutte le <a>, ma solo a quelle che avranno specificato anche l'attributo title

Il **secondo** agisce in base alla presenza di un attributo valorizzato in un determinato modo

```
<style>
      input[type="text"]
</style>
```

In questo esempio la regola CSS non andrà applicata a tutti gli <input>, tanto meno a quelle che presentino anche l'attributo type; sarà necessario, invece che l'attributo type abbia come valore "text".
Questo tipo di selettore è utilizzato molto proprio con i TAG <input> delle <form>

Sintassi di base, testo e font

Il CSS si avvale di cinque proprietà per definire gli stili del testo:

- font-family
- font-style
- font-variant
- font-weight
- font-size

Una sesta proprietà - font - permette di specificare in un'unica volta le proprietà family, size, style, variant e line-height.

Altre due proprietà che non specificano caratteristiche dei font, ma che sono strettamente correlate ad essi, sono:

- text-decoration
  che aggiunge accorgimenti al testo quali, per esempio, la sottolineatura o il "testo barrato";

- text-transform
  che formatta tutto il testo, al quale è applicato, da maiuscolo a minuscolo o viceversa.

Sintassi di base, testo e font

La proprietà **font-family** accetta due tipi di valori:

- il nome di una famiglia di caratteri;
- un generico tipo di carattere (es. Serif).

E' possibile attribuire a questa proprietà più nomi di caratteri separati da una virgola (,)

```
<style>
    body{font-family: Arial, Helvetica, sans-serif}
</style>
```

Può, accadere che un browser non abbia a disposizione nessuno dei caratteri previsti nella proprietà font-family; se si aggiunge un generico tipo di carattere come "sans-serif" alla lista, il browser visualizzerà il testo nel primo carattere di tipo sans-serif disponibile.

I caratteri generici sono:

- serif
- sans-serif
- monospace
- cursive
- fantasy

Sintassi di base, testo e font

Grazie alla proprietà **@font-face** è possibile caricare all'interno delle pagine web font non standard, salvati fisicamente all'interno della directory del sito web.

Va precisato che, purtroppo, non esiste ad oggi una tipologia di font standard per tutti i browser.

Browser	EOT	TTF	WOFF
Mozilla Firefox (<= 3.5)		✓	
Mozilla Firefox (>= 3.6)			✓
Opera		✓	
Chrome		✓	
Safari		✓	
Internet Explorer	✓		

Esempio di codice con @font-face

```
@font-face {
   font-family: 'il_mio_font_name';
   src: url(font_name.eot);
   src: local('font_name'), url('font_name.ttf')
        format('truetype');
}

body{ font-family: il_mio_font_name, Arial, sans-serif}
```

Le due regole src, inserite in questo ordine, evitano inutili richieste HTTP ad Internet Explorer.

Sintassi di base, testo e font

I web fonts con Google Font

Google offre la possibilità di utilizzare centinaia di font *open,* utilizzabili a piacimento su qualsiasi tipo di sito web: https://www.google.com/fonts

Il funzionamento di Google Font Api è davvero molto semplice, infatti Big G. mette a disposizione la propria infrastruttura per cui non è necessario scaricare (in locale o sui nostri server) alcun file, tanto meno dover scrivere alcuna riga di codice.

Di fatto va inserito nella pagina HTML un link in questo modo:

```
<head>
   <link href='https://fonts.googleapis.com/css?family=Lato' rel='stylesheet' type='text/css'>
</head>
```

E richiamarlo nello stile così:

```
<head>
   body{ font-family: 'Lato', Arial, sans-serif}
</head>
```

Sintassi di base, testo e font

La proprietà **font-style** accetta 3 valori:

- normal (default)
- italic
- oblique

Italico e obliquo sono stili molto simili ma non identici (solo nei caratteri di tipo serif si nota una differenza).

Esempi di regole di stile che settano la proprietà font-style sono:

```
h1,h2{font-style: italic}
p{font-style: oblique}
```

valori che possono essere assegnati alla proprietà **font-variant** sono:

- normal
- small-caps

Ovviamente, è il valore small-caps che trasforma il testo in maiuscoletto.
Se non è disponibile il carattere maiuscoletto, il browser tenterà di crearlo riducendo il normale carattere maiuscolo di quel font. Se neppure questa ultima operazione è possibile, verrà utilizzato il carattere maiuscolo senza alcuna riduzione.

```
h1{font-variant: small-caps}
```

Sintassi di base, testo e font

Font-weight:

Ci sono nove livelli di spessore del testo, da 100 a 900; 100, ovviamente, indica lo spessore minore. E' possibile, altresì, specificare valori come "normal" che corrisponde al valore 400 e "bold" che corrisponde al valore 700.

In realtà, ci sono pochissime famiglie di font che possiedono l'equivalente di tutti i 9 livelli di spessore del font; nella maggior parte dei casi, infatti, non si notano differenze, ad esempio, tra lo spessore 100 e 200, tra 200 e 300 oppure tra 800 e 900.

Altri due valori - "bolder" e "lighter" - selezionano uno spessore che è relativo allo spessore dell'elemento di appartenenza

```
h1{font-weight:700}
p{font-weight: bold}
```

Con la proprietà font-size è possibile definire la grandezza del carattere.

Qui è necessario fare, nella slide che segue, una panoramica sulle unità di misura utilizzate nei CSS.

Le unità di misura dei CSS

Tipi di valore
Nei CSS i valori possono essere espressi da:

- **Numeri -** possono essere definiti come numeri interi (1, 10, 25, etc.) o in virgola mobile (1.2, 2.55, 3.80, etc.)
- **unità di misura** (16px, 14pt, 1.855em)
- **percentuali** (50%, 55.5%)
- **codici per la definizione dei colori** (#FFCC00, #000)
- **URI** (background-image: url(immagini/sfondo-pagina.jpg))
- **parole chiave** (background-size: content-box)
- **stringhe di testo** (content: "I love CSS3")

Queste le unità di misura utilizzate per definire dimensioni, spazi e distanze:

Attributo	Descrizione
pt	unità di misura tipografica utilizzata per la dimensione dei font. Il suo utilizzo è di fatto limitato ai CSS per la stampa.
px	unità di misura degli schermi. È quella più usata e facile da applicare
em	unità di misura di ampio utilizzo se si desidera impostare le dimensioni dei font o dei box in maniera **relativa**.
rem	Simile a em, ma con una gestione più efficace, in alcuni casi, delle dipendenze che determinano la relatività
%	Unità di misura che determina il valore inserito come percentuale

Le unità di misura dei CSS

I px e i pt sono unità di misura che permettono di inserire un valore ben preciso.

Al contrario le altre unità di misura lavorano basandosi sulla relatività, ovvero un qualsiasi valore espresso in %, em o rem fa riferimento alla dimensione di un elemento "genitore".

```
<style>
    ol{font-size: 16px}
    ol ol{ font-size:0.8em}
</style>

<body>
    <ol>
        <li>Primo livello
            <ol>
                <li>Secondo livello - Primo elemento</li>
                <li>Secondo livello - Secondo elemento</li>
            </ol>
        <li>Secondo livello</li>
    </ol>
</body>
```

1. Primo livello
    1. Secondo livello - Primo elemento
    2. Secondo livello - Secondo elemento
2. Secondo livello

Sintassi di base, testo e font

La proprietà **text-transform** si può impostare su quattro valori:

- capitalize
  La Prima Lettera Di Ogni Parola Viene Trasformata in Maiuscolo;
- uppercase
  TUTTE LE LETTERE DI TUTTE LE PAROLE VENGONO TRASFORMATE IN MAIUSCOLO
- lowercase
  tutte le lettere di tutte le parole vengono trasformate in minuscolo;
- none
  nessun effetto particolare.

```
h1{text-transform: uppercase}
p{text-transform: capitalize}
```

Sintassi di base, testo e font

Ci sono altre 6 proprietà che agiscono sui contenuti:

- text-align
- text-indent
- line-height
- word-spacing
- letter-spacing
- vertical-align

La proprietà **text-align** accetta 4 valori:

- left - il testo è allineato a sinistra (valore di default)
- right - il testo è allineato a destra
- center - il testo è centrato rispetto all'elemento «genitore»
- justify – il testo è giustificato

La proprietà text-align funziona solo se applicata a elementi di blocco (div, h1, p, ...) ed è una proprietà che gli elementi ereditano dai loro "genitori".

```
h1{text-align: center}
```

Sintassi di base, testo e font

La proprietà **text-indent** agisce sull'indentazione del testo.
Va detto che, sebbene è una pratica molto utilizzata nella stampa tipografica, l'indentazione del testo non è così diffusa nelle pagine web.

text-indent accetta 2 valori

- 25px - valore assoluto;
- 10% - valore percentuale rispetto al contenitore

```
p{text-indent: 25px}
```

La proprietà **line-height** modifica lo spazio occupato in altezza dal testo.
E' molto utilizzato per modificare l'interlinea nei testi, al fine di migliorarne la leggibilità.

line-height accetta 3 valori:

- 25px - valore assoluto;
- 10% - valore percentuale rispetto al contenitore
- 1.4em - valore relativo rispetto al contenitore

```
p{line-height: 1em}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

```
p{line-height: 1.4em}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Sintassi di base, testo e font

Le proprietà **word-spacing** e **letter-spacing** agiscono sulla spaziatura orizzontale tra le parole (word-spacing) e le lettere (letter-spacing)
Come per text-indent anche queste due proprietà vengono utilizzate molto di rado.

Entrambe le proprietà accettano esclusivamente valori assoluti

```
h1{letter-spacing: 5px}
```

IO ADORO I CSS3

```
h1{letter-spacing: 15px}
```

I O A D O R O I C S S 3

La proprietà **vertical-align** agisce sull'allineamento verticale del testo rispetto al proprio contenitore.
A differenza delle altre proprietà accetta però diversi valori.

Suggerisco questo link per osservare tutti i possibili valori assegnabili.

La proprietà display

La proprietà **display** serve per controllare con i CSS il modo in cui un elemento viene reso dal browser.
Attraverso questa proprietà è possibile manipolare gli elementi di blocco, gli elementi in linea, gli elementi di lista e gli elementi di tabella.

La definizione di questa proprietà è molto semplice:

```
selettore{display: valore}
```

Il valore può essere rappresentato unicamente da una delle seguenti parole chiave:

Valore	Descrizione
block	Il TAG viene reso come un elemento blocco, per cui occupa un'intera riga del proprio contenitore
inline	Il TAG a cui viene applicata assume le caratteristiche degli elementi inline
inline-block	Il TAG può assumere, come gli elementi blocco, dimensioni esplicite (larghezza e altezza), margini e padding, ma come tutti gli elementi inline, si disporrà orizzontalmente e non verticalmente, potendo essere circondato dal testo ed essendo sensibile all'allineamento vertical. In pratica si differenzia dal valore block perché non occupa un'intera riga.
none	Il TAG non viene mostrato a video, è come se non fosse nemmeno presente nel documento, in quanto non occupa nessuno spazio visibile.

Ereditarietà, cascata e conflitti

In che modo regole e proprietà interagiscono tra di loro nel contesto di un foglio di stile?
Per spiegare come le regole dei fogli di stile sussistano tra esse e, in un certo qual modo, si ripettino, dobbiamo spiegare alcuni conetti molto importanti.

Il primo concetto è quello di **ereditarietà**.
Secondo questo meccanismo le impostazioni di stile applicate a un elemento vengono ereditate anche dai suoi discendenti.
Almeno fino a quando, per un elemento discendente, non si imposti esplicitamente un valore diverso per quella proprietà.

Se per esempio impostiamo il colore rosso scuro per il testo a livello dell'elemento <body>:

```
body {color: red;}
```

Tutti gli elementi discendenti di <body>, erediteranno questa impostazione. Ma se ad un certo punto definiamo nel codice del CSS un selettore con la proprietà color: green; l'ereditarietà viene interrotta:

```
body {color: red;}
li {color: green;}
```

Gli elementi <li> avranno, a differenza degli altri, il testo verde.

Non tutte le proprietà sono però ereditate e lo renderemo esplicito nell'analisi di ciascuna di esse. In genere le proprietà non ereditate sono quelle attinenti la formattazione del box model: margini, bordi, padding, background le più importanti.
Il motivo è semplice: ereditare un bordo, per esempio, è semplicemente senza senso.
Se ne imposto uno per un , sarebbe assurdo che i TAG <a> in esso contenuto vengano circondati dallo stesso bordo!

Ereditarietà, cascata e conflitti

Peso e origine
Come posso gestire i conflitti possibili tra gli stili e le regole?
Facciamo un esempio.

```
<style>
    p{color: black;}
    .testo{color: white;}
</style>
<body>
    <p class="testo">Testo del paragrafo</p>
</body>
```

Perché in questo esempio il testo del paragrafo sarà bianco e non nero?
Perché il selettore di classe prevale su quello di tipo con l'elemento?

La **specificità**, come il nome può suggerire, descrive il **peso** relativo delle varie regole all'interno di un foglio di stile. Esistono regole ben precise per calcolarla e sono quelle che applica un browser quando si trova davanti ad un CSS.

Tutti i browser web consentono una gestione di questo aspetto.

Esiste una gerarchia nel modo in cui vengono scritte le regole CSS e quindi vengono interpretate dal Browser.

Questa gerarchia, vedremo in alcuni esempi, può essere modificata con l'uso della parola chiave **!important**.
Va detto, comunque, che un foglio di stile CSS ben strutturato non avrà mai bisogno di utilizzare !important *per forzare la mano*.

Ereditarietà, cascata e conflitti

La specificità, come il nome può suggerire, descrive il peso relativo delle varie regole all'interno di un foglio di stile. Esistono regole ben precise per calcolarla e sono quelle che applica un Browser quando si trova davanti ad un CSS.
I fattori del calcolo sono 3 e ciascuno di essi rappresenta il valore di una tripletta. Per prima cosa si conta il numero di selettori id presenti nella regola. Si passa quindi a verificare la presenza di classi e pseudo-classi. Infine si conta il numero di elementi definiti nella regola.

Prima regola

```
#titolo{color: black;}
```

Calcolo: 1 id, 0 classi, 0 elementi. Tripletta dei valori: 1-0-0

Seconda regola

```
.classe{background: #C00;}
```

Calcolo: 0 id, 1 classe, 0 elementi. Tripletta dei valori : 0-1-0

Terza regola

```
h1{color: red;}
```

Calcolo: 0 id, 0 classe, 1 elementi. Tripletta dei valori : 0-0-1

Il peso specifico della prima regola è il maggiore. Quello dell'ultima il minore.
In pratica: **gli id pesano più delle classi che pesano più dei singoli elementi**.
Non commettete l'errore di valutare il numero più grande a prescindere dalla sua posizione.
La regola che segue presenta la specificità 1-0-0: `#paragrafo {color: green;}`
ed è più importante di questa che ha i valori 0-0-2: `div p {color: red;}`

Ereditarietà, cascata e conflitti

Il concetto di cascata
Cerchiamo di riassumere adesso il concetto e il meccanismo della cascata ricostruendo il procedimento di un browser quando incontra un foglio di stile e lo rende sul monitor del nostro computer:

1. Per prima cosa controlla il target stabilito con l'attributo media o con dichiarazioni equivalenti.
2. Scarta tutti gli stili riferiti alla stampa o ad altri supporti. Allo stesso tempo scarta tutte le regole che non trovino corrispondenza negli elementi strutturali del documento.
3. Comincia ad ordinare per peso e origine secondo le regole viste.
4. Quindi calcola la specificità dei selettori e, in caso di conflitto tra regole, usa questo criterio di prevalenza.
5. Se non ci sono conflitti o se peso, origine e specificità coincidono, viene applicata la regola più vicina all'elemento nel codice del documento. L'ordine, se le dichiarazioni degli stili sono fatte nell'ordine più corretto e logico, è quindi il seguente: **gli stili in linea prevalgono su quelli incorporati che a loro volta prevalgono su quelli collegati**.

Semplice e lineare la regola dell'**importanza**: se una dichiarazione viene accompagnata dalla parola chiave !important essa balza al primo posto nell'ordine di applicazione a prescindere da peso, origine, specificità e ordine.

```
<style>
    p{color: black !important;}
    .testo{color: white;}
</style>
<body>
    <p class="testo">Testo del paragrafo</p>
</body>
```

In questo esempio, benché <p> presenti una classe è più forte del selettore di tipo, grazie alla parola chiave !important, la regola più debole vincerà sulla più forte.
Come già detto, non è una buona abitudine riempire i fogli di stile con questa parola chiave!

Il box model

Se diciamo che un paragrafo o un titolo possono essere pensati come una "scatola", possiamo facilmente disegnarli, come in figura, come del testo racchiuso da un immaginario rettangolo.
Pensando ai TAG in questo modo è possibile realizzare le strutture che permettono l'organizzazione a video dei contenuti web.

```
<body>
    <h1>I love css3</h1>
</body>
```

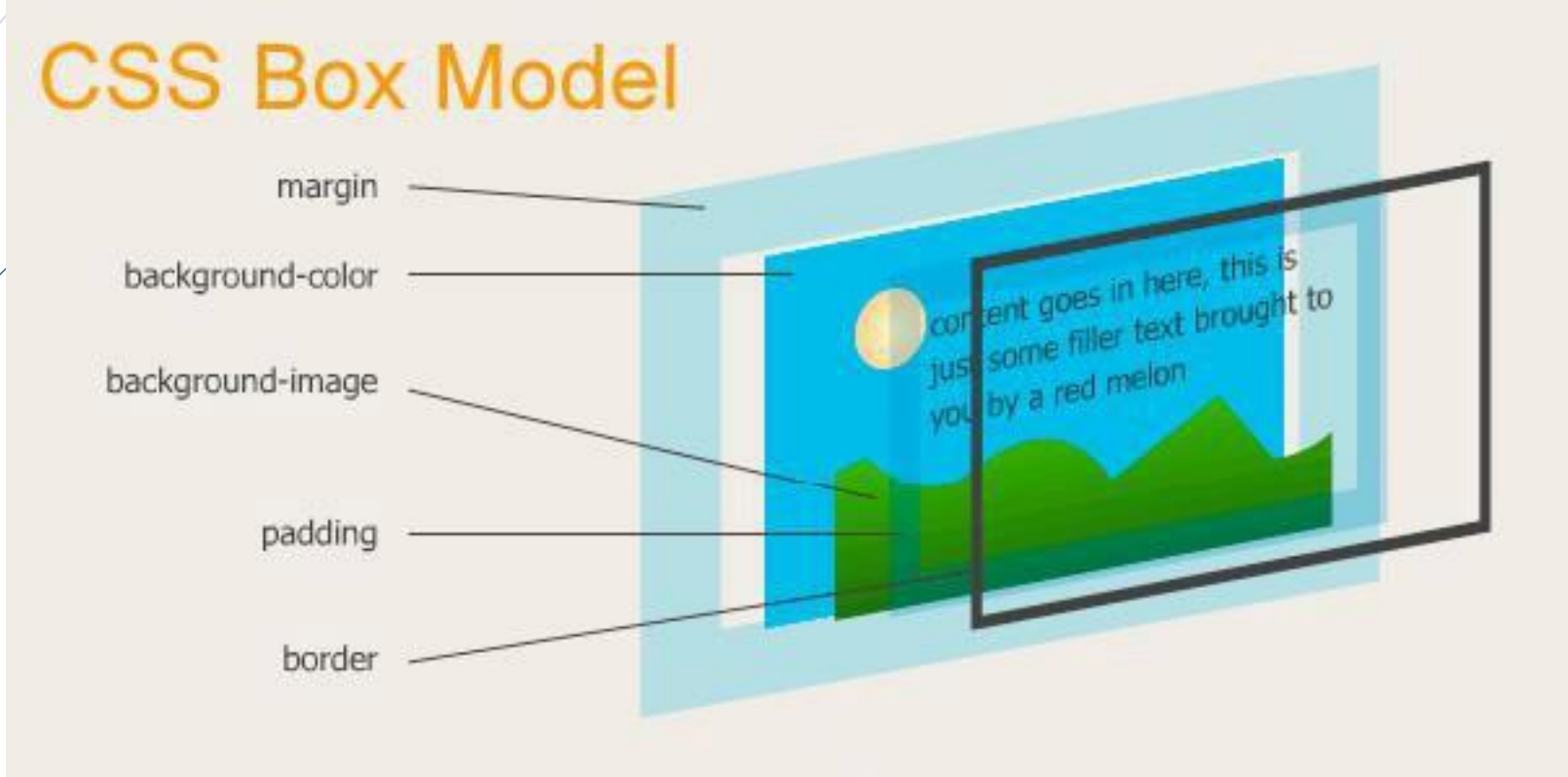
Ci sono tre "zone" all'*esterno del contenuto* che possono essere manipolate con il CSS:

- margin
- padding
- border

Altre proprietà opzionali che permettono un migliore "aggiustamento" degli spazi dei vari elementi sono:

- width
- height
- float
- clear

Il box model


CSS Box Model
margin
background-color
background-image
padding
border
by a red melon

Il box model

Il **margin** è lo **spazio esterno** al TAG di riferimento. Serve a distanziare un elemento HTML da un altro

Ci sono cinque proprietà che definiscono i margini:

- margin-left (margine sinistro);
- margin-right (margine destro);
- margin-top (margine superiore);
- margin-bottom (margine inferiore);
- margin (proprietà che permette di impostare tutti i quattro margini in una volta sola).

Si possono impostare le proprietà dei margini con tre possibili tipi di valore:

- valore assoluto - es. 10pt;
- valore percentuale – es. 2% della larghezza dell'elemento "genitore";
- valore "auto" - valore automatico che può essere applicato solo per i margini sinistro e destro.

```
<style>
    h1{margin:10px}
</style>
```

Il box model

Utilizzando la proprietà **margin** bisogna, però, fare attenzione a:

- i valori vengono interpretati come top/right/bottom/left;
- se si inserisce un solo valore, questo viene utilizzato per tutti i 4 margini;
- se sono impostati due o tre valori, vale la regola che il valore mancante verrà posto uguale al suo opposto.

Esempi:

`h1{margin: 2em}`	tutti i margini saranno impostati a 2em
`h1{margin: 1em 2em}`	i margini superiore e inferiore saranno impostati a 1em, i margini sinistro e destro saranno impostati a 2em
`h1{margin: 1em 2em 3em}`	il margine superiore sarà impostato a 1em, i margini sinistro e destro saranno impostati a 2em, il margine inferiore sarà impostato a 3em
`h1{margin: 1em 2em 3em 4em}`	i valori verranno interpretati come superiore 1em, destro 2em, inferiore 3em e sinistro 4em

Il box model

La proprietà "**padding**" definisce **lo spazio tra il perimetro del TAG e il proprio contenuto**.

Per comprendere meglio di cosa si tratta potremmo pensare al padding come al *passe-partout* che viene inserito nei quadri per separare la cornice dalla tela.

Queste le 5 proprietà che definiscono il padding:

- padding - left (spazio a sinistra del TAG);
- padding - right (spazio a destra del TAG);
- padding - top (spazio superiore del TAG);
- padding - bottom (spazio inferiore del TAG);
- padding (proprietà che permette di impostare tutti i quattro lati del TAG in una volta sola).

```
<style>
    h1{padding:10px}
</style>
```

Il box model

Utilizzando la proprietà **padding** bisogna, così come per il margin, fare attenzione a:

- i valori vengono interpretati come top/right/bottom/left;
- se si inserisce un solo valore, questo viene utilizzato per tutti i 4 lati;
- se sono impostati due o tre valori, vale la regola che il valore mancante verrà posto uguale al suo opposto.

Esempi:

`h1{padding: 2em}`	tutti gli spazi tra TAG e contenuto saranno impostati a 2em
`h1{padding: 1em 2em}`	Gli spazi superiore e inferiore saranno impostati a 1em, gli spazi sinistro e destro saranno impostati a 2em
`h1{padding: 1em 2em 3em}`	Lo spazio superiore sarà impostato a 1em, gli spazi sinistro e destro saranno impostati a 2em, lo spazio inferiore sarà impostato a 3em
`h1{padding: 1em 2em 3em 4em}`	i valori verranno interpretati come superiore 1em, destro 2em, inferiore 3em e sinistro 4em

Il box model

Un'altra proprietà utilizzata nel box-model è il **border**: serve a definire la *cornice* dei TAG

Ci sono 12 proprietà che definiscono lo stile del bordo; le prime 5 (fondamentali) sono:

- border-left (bordo sinistro);
- border-right (bordo destro);
- border-top (bordo superiore);
- border-bottom (bordo inferiore);
- border (proprietà che permette di impostare tutti i quattro lati del bordo in una volta sola).

Altre proprietà per definire particolari stili del bordo sono:

- border-color (colore del bordo);
- border-style (stile del bordo);
- border-left-width (spessore del bordo sinistro);
- border-right-width (spessore del bordo destro);
- border-top-width (spessore del bordo superiore);
- border-bottom-width (spessore del bordo inferiore);
- border-width (spessore dei 4 lati del bordo).

Il box model

border-color è la proprietà che definisce il **colore** del bordo.

Può essere espressa usando:

- nomi predefiniti - es. blue, red, ...;
- la notazione esadecimale - es. #3A6C98
- la notazione RGB - es. rgb(255,255,255).

* *Vedremo nelle slide successive come vengono gestiti i colori nei CSS.*

Bisogna, però, fare attenzione a:

- i valori vengono interpretati come top/right/bottom/left;
- se si inserisce un solo valore questo viene utilizzato per tutti i 4 lati;
- se sono impostati due o tre valori vale la regola che il valore mancante verrà posto uguale al suo opposto.

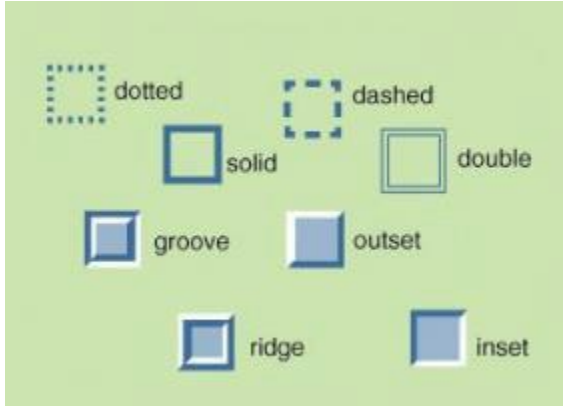
Se non si specifica alcun colore, il bordo (se c'è) sarà del colore dell'elemento genitore.

Il box model

border-style è la proprietà che definisce la **tipologia** di bordo che verrà disegnato a video.

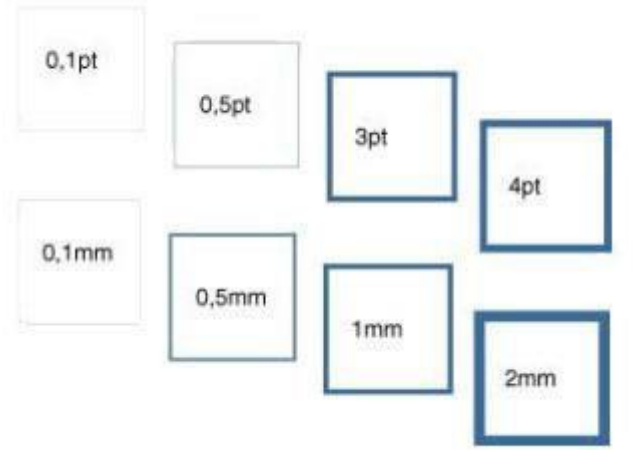
Questi sono i valori che accetta la proprietà border-style:

- none - nessun bordo è disegnato, a prescindere dallo spessore definito nella proprietà border-width;
- dotted - un bordo tratteggiato "a puntini";
- dashed - un bordo tratteggiato "a linee";
- solid - un bordo che è una linea continua;
- double - una doppia linea (la somma dello spessore delle due linee e dello spazio che intercorre tra esse è pari al valore espresso nella proprietà border-width);
- groove - un bordo "scavato" nello sfondo;
- ridge - un bordo "in rilievo" rispetto alla sfondo;
- inset - un bordo che dà un effetto "scavato" all'elemento che delimita;
- outset - un bordo dà un effetto "in rilievo" all'elemento che delimita.



Il box model

border-width è la proprietà che definisce lo **spessore** del bordo.



Anche le proprietà border-width sono 5:
- border-left-width (spessore del bordo sinistro);
- border-right-width (spessore del bordo destro);
- border-top-width (spessore del bordo superiore);
- border-bottom-width (spessore del bordo inferiore);
- border-width (permette di impostare lo spessore del bordo in una volta sola su tutti i lati).

Questi sono i valori che accetta la proprietà Border-width:
- thin – sottile // medium - medio (default) // thick - spesso;
- un valore assoluto (3px);
- un valore percentuale (1%);
- none - spessore 0.

Se si usano le parole chiave thin, medium o thick l'effetto finale e anche le differenze di spessore (o la non differenza) dipende dal browser che sta visualizzando la pagina.

Anche in questo caso valgono gli accorgimenti visti in precedenza:
- i valori vengono interpretati come top/right/bottom/left;
- se si inserisce un solo valore questo viene utilizzato per tutti i 4 lati;
- se sono impostati due o tre valori vale la regola che il valore mancante verrà posto uguale al suo opposto

Il box model

A differenza di quanto accade per "margin" e "padding", però, utilizzando la proprietà "border" non si possono settare stili diversi per i quattro lati del bordo.

Ad esempio la regola: p{border: solid red}

definisce uno stile di linea continua verde e di spessore "medium" (non essendo dichiarato si assume il valore di default che è medium) per tutti i quattro lati del bordo a tutti i paragrafi (<p>).

Volendo definire stili diversi per i quattro lati, bisogna utilizzare le proprietà descritte in precedenza.

L'ordine in cui si definiscono width, style e color non è importante.

Gli esempi che seguono sono perfettamente equivalenti:

```
p{border: solid red 2px}

p{border: 2px solid red}

p{border: red 2px solid}
```

Il box model

border-radius è una proprietà dei CSS3 che permette di realizzare bordi arrotondati.
E' supportata da tutti Browser. Solo IE Explorer l'ha implementata dalla versione 9.
Nel caso si utilizzi IE8, il bordo verrà mostrato con gli angoli non arrotondati.

Anche le proprietà border-radius sono 5:

- border-top-left-radius (agisce sul bordo in alto a sinistra)
- border-top-right-radius (agisce sul bordo in alto a destra)
- border-bottom-left-radius (agisce sul bordo in basso a sinistra)
- border-bottom-right-radius (agisce sul bordo in basso a destra)
- border-radius (agisce in modo uguale su tutti i bordi)

```
p{
    border: solid black 5px;
    border-radius:20px;
}
```

I valori della proprietà border-radius possono essere espressi:

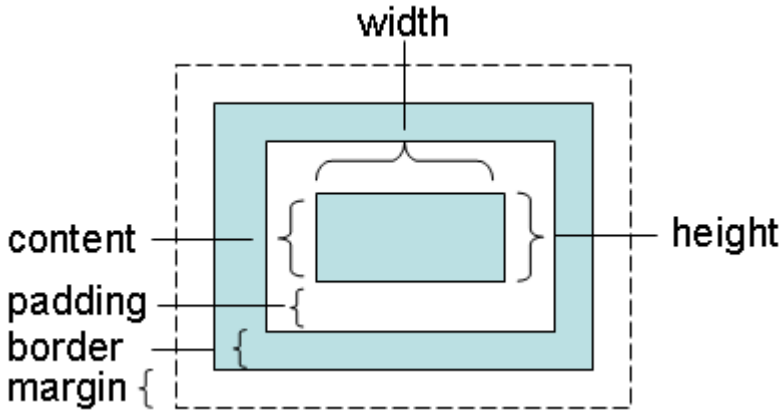
- 10px (valore assoluto);
- 5% (valore in percentuale);
- none – nessun arrotondamento (default).

Il box model

L'altezza di un elemento è determinata dal suo contenuto.
Più testo inserisco in box più esso sarà esteso in senso verticale.

Esiste comunque una proprietà che permette di settare arbitrariamente l'altezza di un TAG: **height**.
Questa proprietà definisce la distanza tra il bordo superiore e quello inferiore di un elemento.

```
p{ height: 100px; }
```



Il valore può essere espresso da:

- 10px (un valore numerico con unità di misura);
- 30% (un valore in percentuale)
  Attenzione: il valore in percentuale è sempre definito rispetto all'altezza del blocco contenitore, purché esso abbia un'altezza esplicitamente dichiarata; diversamente, la percentuale viene interpretata come auto;
- auto: l'altezza sarà quella determinata dal contenuto (valore di default).

Il box model

L'altezza può essere gestita utilizzando altre 2 proprietà:

- **min-height** (imposta l'altezza minima per un elemento)
- **max-height** (imposta l'altezza massima per un elemento)

La tipologia di valori espressi è la stessa di height:

- 10px (un valore numerico con unità di misura);
- 30% (un valore in percentuale);
- none (non specifico nessun valore).

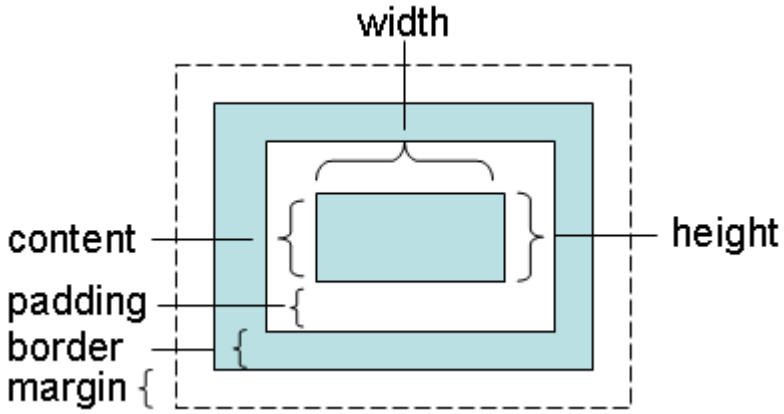
```
p{min-height:100px; }
p{max-height:500px;}
```

Il box model

La larghezza di un elemento (**width**) può essere in alcuni casi determinata dal suo contenuto, in altri (di default) è il 100% dello spazio a disposizione.

La differenza tra questi casi è la vedremo più avanti quando analizzeremo la proprietà display.

```
p{ width: 500px; }
```



Il valore può essere espresso da:

- 10px (un valore numerico con unità di misura);
- 30% (un valore in percentuale che sarà calcolato rispetto alla larghezza dell'elemento contenitore);
- auto: la larghezza sarà determinata dal contenuto (possono esserci delle eccezioni gestite dalla proprietà display).

Il box model

Anche la larghezza può essere gestita utilizzando altre 2 proprietà:

- **min-width** (imposta la larghezza minima per un elemento)
- **max-width** (imposta la larghezza massima per un elemento)

La tipologia di valori espressi è la stessa di width:

- 10px (un valore numerico con unità di misura);
- 30% (un valore in percentuale);
- none (non specifico nessun valore).

```
p{min-width:100px; }
p{max-width:50%;}
```

Il box model

Strettamente collegata alle proprietà height e width è la proprietà **overflow**.

Essa fornisce un modo per gestire il contenuto che superi i limiti impostati manualmente.
Serve infatti per definire il comportamento di un elemento blocco nel caso il suo contenuto ecceda dalle sue dimensioni dichiarate, ossia nel caso in cui l'area utile del box non sia sufficiente per i contenuti.
Si applica a tutti gli elementi blocco e non è ereditata.

I valori possono essere espressi con le parole chiave:

- visible: valore iniziale, il contenuto eccedente rimane visibile;
- hidden: il contenuto eccedente non viene mostrato;
- scroll: il browser crea barre di scorrimento che consentono di fruire del contenuto eccedente;
- auto: il browser tratta il contenuto eccedente secondo le sue impostazioni predefinite; di norma dovrebbe mostrare una barra di scorrimento laterale.

```
p{
   height: 200px;
   overflow: auto;
}
```

Il box model

Correlate a overflow esistono anche altre 2 proprietà:

- **overflow-x** (per gestire il contenuto che superi i limiti impostati sulla proprietà width)
- **overflow-y** (per gestire il contenuto che superi i limiti impostati sulla proprietà height)

I valori espressi sono gli stessi usati con overflow:

- visible: valore iniziale, il contenuto eccedente rimane visibile;
- hidden: il contenuto eccedente non viene mostrato;
- scroll: il browser crea barre di scorrimento che consentono di fruire del contenuto eccedente;
- auto: il browser tratta il contenuto eccedente secondo le sue impostazioni predefinite; di norma dovrebbe mostrare una barra di scorrimento laterale.

```
p{
    width:500px;
    height: 200px;
    overflow-x: hidden;
    overflow-y: auto;
}
```

In questo esempio la scroll-bar orizzontale non sarà mai visibile, mentre quella verticale sarà visibile solo nel caso il contenuto superi i limiti imposti nella proprietà height

Il box model

Il box model dei CSS permette di definire dei blocchi rettangolari con specifici valori per la larghezza e l'altezza della sezione Contenuti, del padding, dei bordi, dei margini.

La larghezza del box fino al bordo è la somma della larghezza specificata con width + ampiezza del padding + spessore del bordo + eventuale margin aggiunto.

Stessa cosa accade per l'altezza di un box.

Si consideri il seguente codice CSS:

```
.box {
   width:200px;
   border:5px solid #900;
   padding:7px;
   margin:10px
}
```

Il risultato di questo esempio sarà un box
largo 200px + width [2 x 5px] + padding [2 x 7px] + margin [2 x 10px] = 244px.

La larghezza complessiva è data dai 200px dei contenuti a cui devono essere sommati 5px per ognuno dei bordi laterali, 7px + 7px per il padding e 10px + 10px per il margin.

Il box model

Modificare il calcolo delle dimensioni con box-sizing

I CSS3 hanno introdotto un modo per modificare il comportamento standard del box model.
Possiamo farlo attraverso la proprietà box-sizing.

```
.box  {
   width:200px;
   border:5px solid #900;
   padding:7px;
   margin:10px
   box-sizing: border-box;
}
```

In questo box la larghezza impostata con la proprietà width verrà calcolata **inglobando i valori per il padding e i bordi** (escluso il margin), per cui la larghezza totale occupata dal box sarà di 220px: 200 width + (2x10) margin

Il box model

Affiancare i box

I div, così come i nuovi TAG semantici HTML5 (header, footer, aside, …), nascono per occupare per intero la riga del documento HTML, per cui non è possibile, se non tramite una proprietà CSS, che essi possano essere disposti a video uno di fianco all'altro.
Per affiancare i TAG di blocco la proprietà CSS usata è **float**.
Spesso nei documenti HTML, oltre al float, viene utilizzata la proprietà **clear**.

Con questa proprietà è possibile rimuovere un elemento dal *normale flusso del documento* e spostarlo su uno dei lati (destro o sinistro) del suo elemento contenitore.
Potemmo dire che il TAG venga *sollevato* dal documento per appoggiarsi a destra o a sinistra (dipende dal valore assegnato al float).

I valori assegnabili alla proprietà float sono:

- left: sposta l'elemento verso sinistra;
- right: sposta l'elemento verso destra;
- none: non applica, o annulla, la proprietà float (valore di default);

```
.box  { float: left}
```

Il box model

Affiancare i box
La proprietà clear serve a impedire che al fianco di un elemento compaiano altri elementi con il float.

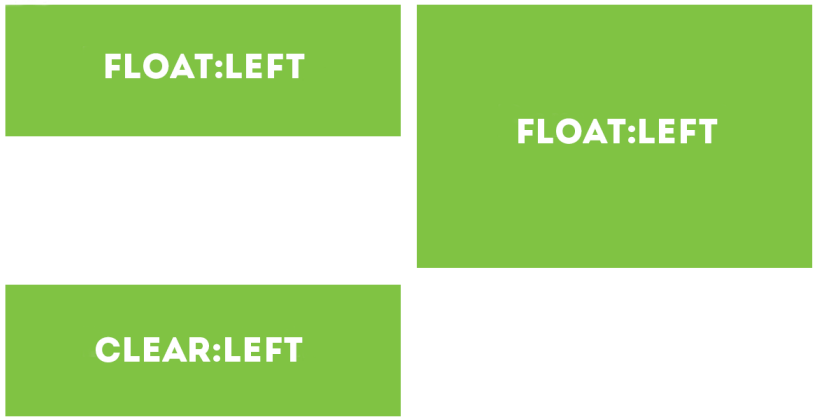
Ma a cosa serve in pratica questa proprietà?
Poiché che il float sposta un elemento dal flusso normale del documento, è possibile che esso venga a trovarsi in posizioni non desiderate, magari al fianco di altri elementi che vogliamo invece tenere separati.

La proprietà clear risolve questo problema.

I valori assegnabili alla proprietà clear sono:

- left: si impedisce il posizionamento a sinistra;
- right: si impedisce il posizionamento a destra;
- both: si impedisce il posizionamento su entrambi i lati;
- none: annulla la proprietà clear (valore di default).

```
.box  { clear: both}
```



Le immagini e i CSS

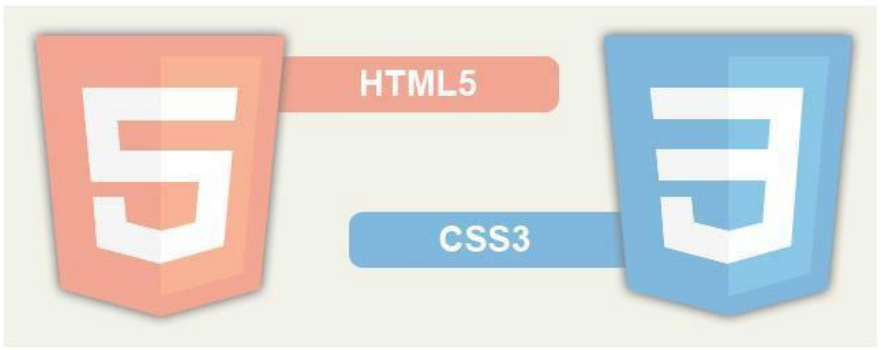
Con i CSS, così come a qualsiasi TAG di blocco, anche alle immagini è possibile aggiungere un border o un margin.
Allo stesso modo è possibile renderle *fluttuanti* e posizionarle a proprio piacimento all'interno dei contenitori genitori.

La sintassi dei CSS non subisce alcuna modifica rispetto a quanto già visto in precedenza.

Una proprietà che viene utilizzata con le immagini è **opacity**.
Serve a gestire il livello di trasparenza dell'elemento a cui viene applicata. Per cui, oltre che per le immagini, la proprietà opacity è applicabile a qualsiasi altro TAG.
Accetta un valore da **0** a **1** (valore di default), che può essere scritto con due cifre dopo la virgola – ad es. 0.2 o 0.25

```
img { opacity:1}
```


```
img { opacity:0.5}
```



I colori nei CSS

Nei file HTML e CSS la gestione dei colori è demandata al modello RGB.
Tale modello di colori si basa sulla miscelazione dei tre colori rosso (Red), verde (Green) e blu (Blue), da cui appunto il nome RGB.
Nel modello RGB, ciascuno dei tre colori è rappresentato da un valore che va da 0 a 100%, dove 100% rappresenta la massima concentrazione del colore.

I valori sono indicati come una tripletta di numeri dove il primo è la concentrazione del colore rosso, il secondo la concentrazione del colore verde e la terza rappresenta la concentrazione del colore blu.

- Il nero è rappresentato dalla tripletta (0,0,0) - assenza di colore
- Il bianco è rappresentato da (100%, 100%, 100%) - presenza di tutti i colori.
- Se abbiamo delle triplette formate dallo stesso numero - es. (40%, 40%, 40%) oppure (90%, 90%, 90%) - abbiamo varie tonalità di grigio.

4 sono i metodi per specificare i colori nei CSS:

- **colori predefiniti** (red, green, yellow, black, … - lista dei colori predefiniti) ;
- **numerazione esadecimale**, caratterizzata da una tripletta RRGGBB preceduta dal simbolo # (ad es. #33CC00).
- **modello RGB**, che può essere definito da una tripletta di numeri da 0 a 255, oppure in percentuale da 0 a 100%
  Ad esempio: rgb(100,250,40) oppure rgb(55%,95%,15%)
- **modello RGBA**, che può essere definito da una tripletta di numeri da 0 a 255, oppure in percentuale da 0 a 100%, con l'aggiunta di un valore da 0 a 1 che determina la trasparenza.
  Ad esempio: rgba(100,50,240,0.75)

I colori nei CSS

Con i CSS è possibile colorare:

- Il **bordo** di un box - con la proprietà **border-color**
- Il **colore del testo** - con la proprietà **color**
- Il **colore di sfondo** di un qualsiasi TAG di blocco (h1, div,) - con la proprietà **background-color**

Il valore di queste proprietà può essere assegnato attraverso una qualsiasi delle modalità viste nella slide precedente.

Un esempio, con la stessa tonalità di colore, scritta nelle diverse modalità:

```
p { color : DeepPink }
```

IO ADORO I CSS3

```
p { color : #ff1493}
```

IO ADORO I CSS3

```
div {border: solid 5px rgb(255,20,147)}
```

```
div {background-color: rgba(255,20,147,1)
```

```
div {background-color: rgba(255,20,147,0.5)
```

Il background

Lo **sfondo di un elemento** può essere gestito attraverso i CSS.
Esso può essere composto da un **colore** o da un'**immagine**.
Se si desidera può anche essere composto anche in parte da un'immagine e in parte da un colore.
Può esserne gestita la trasparenza.
Si può, altresì, stabilire la posizione di un'immagine di sfondo e se essa deve essere ripetuta o meno su tutta la pagina, solo in orizzontale o solo in verticale.

Le proprietà che possono essere modificate al fine di realizzare tali effetti sono:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position
- background-clip
- background-origin
- background-size
- background

Il background

La proprietà **background-color** accetta due valori:

- un colore;
- la parola chiave transparent (default – di fatto non viene mai scritta).

Se si specifica un colore, questo sarà visibile sotto il testo di un elemento.

Se si specifica un colore di sfondo per un elemento di tipo "inline" (es.), esso sarà visibile sotto l'elemento e nello spazio di "padding" intorno ad esso.
Se l'elemento viene visualizzato su più righe, si noterà un effetto "evidenziazione" che segue il testo sulle varie righe.

Il colore di sfondo degli elementi "block" (es. <p>) produce un'area colorata rettangolare (che include anche spazi e righe vuote) intorno all'elemento stesso.

Background su elemento inline

Lorem ipsum dolor sit amet, consectetur *adipiscing elit*, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Background su elemento inline (su più righe)

Lorem ipsum dolor sit amet, consectetur *adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore* magna aliqua.

Background su elemento block

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Il background

La proprietà **background-image** serve a mostrare un'immagine come sfondo dell'elemento.
Accetta 2 tipi di valore:

- **URL** - path assoluto o relativo dell'immagine;
- la parola chiave "none" (default) - nessuna immagine di sfondo.

Quando si specifica un'immagine di sfondo, si può anche impostare un valore per la proprietà background-color: il risultato sarà un colore di sfondo su cui verrà visualizzata anche l'immagine.

Tre sono le ragioni per cui a volte si definiscono dei valori per entrambe le proprietà:

1. il colore sarà visualizzato mentre il browser carica l'immagine (la seconda operazione, solitamente, è più lunga);
2. il colore può essere usato per riempire eventuali parti trasparenti dell'immagine;
3. il colore può essere utilizzato come sfondo nel caso in cui l'immagine non può essere caricata.

```
div {
    background-image: url(immagini/sfondo.jpg);
    background-color:#FF0055;
}
```

Il background

La proprietà **background-repeat** serve a gestire la *ripetizione dell'immagine*, per questo motivo può essere usata solo se nello stesso TAG è presenta la proprietà background-image.
Di default, se l'area del TAG da riempire è più grande dell'immagine caricata, il browser ripeterà la stessa immagine più volte, fino a riempire completamente l'area.
background-repeat serve a modificare questa impostazione di default.

Accetta i seguenti valori (parole chiave):

- repeat -- l'immagine è ripetuta sia orizzontalmente che verticalmente fino a riempire l'intero elemento (default);
- repeat-x -- l'immagine è ripetuta orizzontalmente fino a riempire l'elemento da sinistra a destra ma su una sola riga;
- repeat-y -- l'immagine è ripetuta verticalmente fino a riempire la pagina ma su una sola colonna;
- no-repeat -- l'immagine non è ripetuta, ma visualizzata una sola volta nell'angolo superiore sinistro (a meno che non sia stato impostato un valore diverso nella proprietà background-position).

```
div {
    background-image: url(immagini/sfondo.jpg);
    background-repeat: no-repeat;
}
```

In questo caso, se l'immagine di sfondo è minore dell'area del TAG, non sarà comunque ripetuta dal Browser e sarà posizionata automaticamente in alto a sinistra.

Il background

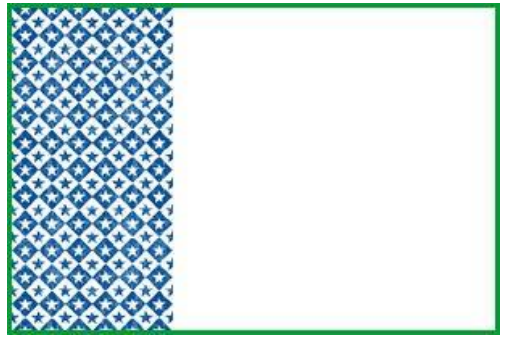
Proponiamo qui alcuni esempio con **background-repeat**

Qui accanto l'immagine che useremo per il nostro sfondo

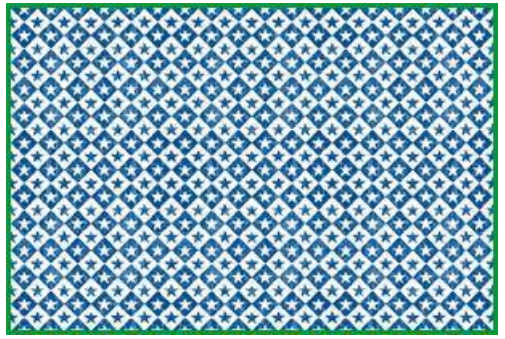
```
div {
   background-image: url(image.jpg)
   background-repeat: no-repeat;
}
```

```
div {
background-image: url(image.jpg)
background-repeat: repeat-x;
}
```

```
div {
background-image: url(image.jpg)
background-repeat: repeat-y;
}
```



```
div {
background-image: url(image.jpg)
background-repeat: repeat;
}
```



Il background

La proprietà **background-attachment**, così come la proprietà background-repeat, può essere assegnata solo in presenza della proprietà background-image.
background-attachment determina se l'immagine è fissa o se deve scorrere all'interno dell'elemento al quale è assegnata.

Accetta questi 2 valori:

- scroll - l'immagine scorrerà sulla finestra insieme al resto (default);
- fixed - l'immagine sarà fissa rispetto alla finestra.

```
div {
    background-image: url(immagini/sfondo.jpg);
    background-attachment: fixed;
}
```

Il background

Come per le due proprietà appena viste, anche **background-position** può essere utilizzata solo in presenza di background-image. background-position serve a gestire il posizionamento dell'immagine di sfondo rispetto al proprio TAG contenitore.

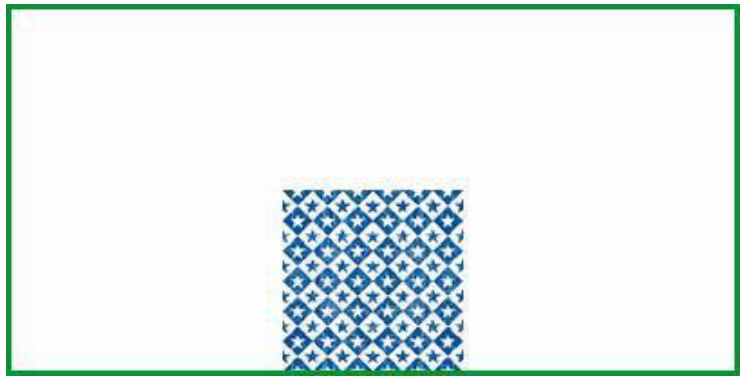
Per utilizzare la proprietà background-position è necessario **specificare 2 valori**:

1. il primo determina il posizionamento dell'immagine sull'asse delle x (orizzontale)
2. il secondo determina il posizionamento dell'immagine sull'asse delle y (verticale)

I valori che si possono associare sono di 3 tipi:

- 20% - valore in percentuale
- 200px - valore assoluto
- center - parola chiave (left, right, center, top, bottom)

```
div {
      background-image: url(immagini/sfondo.jpg);
      background-repeat: no-repeat;
      background-position: center bottom;
}
```



Il background

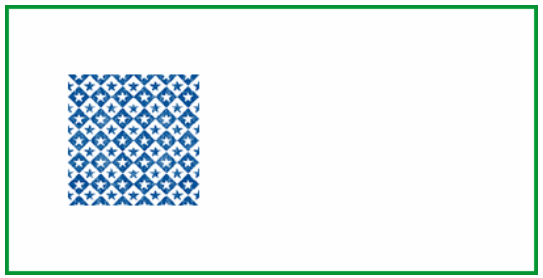
I 2 valori associati alla proprietà background-position possono anche essere di tipi diversi.

E' possibile, per esempio, scrivere un codice come questo:

```
div {
	background-image: url(immagini/sfondo.jpg);
	background-repeat: no-repeat;
	background-position: center 25px;
}
```

Nel caso in cui, invece, venisse indicato un solo valore, come questo in questo esempio

```
div {
	background-image: url(immagini/sfondo.jpg);
	background-repeat: no-repeat;
	background-position: 15%;
}
```



esso sarà assegnato per il posizionamento orizzontale dell'immagine, mentre il secondo sarà ignorato e sostituito dal valore di default previsto per il posizionamento verticale (center).

Il background

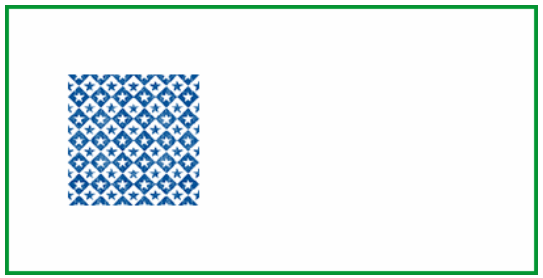
I 2 valori associati alla proprietà background-position possono anche essere di tipi diversi.

E' possibile, per esempio, scrivere un codice come questo:

```
div {
	background-image: url(immagini/sfondo.jpg);
	background-repeat: no-repeat;
	background-position: center 25px;
}
```

Nel caso in cui, invece, venisse indicato un solo valore, come questo in questo esempio

```
div {
	background-image: url(immagini/sfondo.jpg);
	background-repeat: no-repeat;
	background-position: 15%;
}
```



esso sarà assegnato per il posizionamento orizzontale dell'immagine, mentre il secondo sarà ignorato e sostituito dal valore di default previsto per il posizionamento verticale (center).

Il background

background-clip è una proprietà che può essere utilizzata solo in presenza di un padding maggiore di 0 e dalla proprietà background-color.
Serve a specificare se il colore di sfondo debba coprire tutta l'area del TAG (padding compreso), o se deve coprire solo l'area del contenuto (padding escluso).

Accetta solo 2 valori (parole chiave):

- border-box -- indica che lo sfondo copre anche l'area occupata dal padding (valore di default)
- content-box -- indica che lo sfondo copre solo l'area del contenuto (padding escluso)

```
div {
    width: 400px;
    padding:20px;
    background-color:#f9b234;
    background-clip: border-box;
}
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

```
div {
    width: 400px
    padding: 20px;
    background-color: #f9b234;
    background-clip: content-box;
}
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

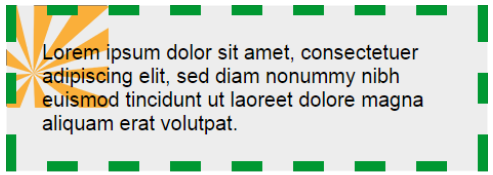
Il background

background-origin è l'*alter ego* di background-clip, l'unica differenza sta che viene associata al background-image, mentre background-clip viene associata a background-color.

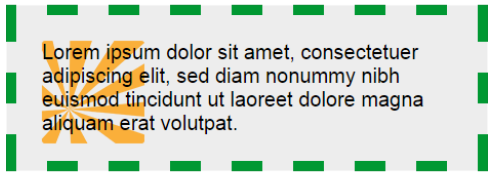
Accetta 3 valori (parole chiave):

- padding-box -- indica che l'immagine di sfondo copre l'area occupata anche dal padding, ma non dal bordo (valore di default)
- border-box -- indica che l'immagine di sfondo copre l'area occupata anche dal padding e dal bordo
- content-box -- indica che l'immagine di sfondo copre solo l'area del contenuto (padding e bordo esclusi)

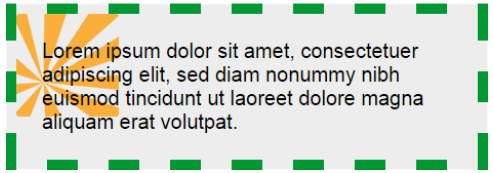
```
div {
    padding:20px;
    border: dashed 5px #009933;
    background-image: url(image.jpg);
    background-clip: border-box;}
```




```
div {
    padding:20px;
    border: dashed 5px #009933;
    background-image: url(image.jpg);
    background-clip: content-box;}
```




```
div {
    padding:20px;
    border: dashed 5px #009933;
    background-image: url(image.jpg);
    background-clip: padding-box;}
```



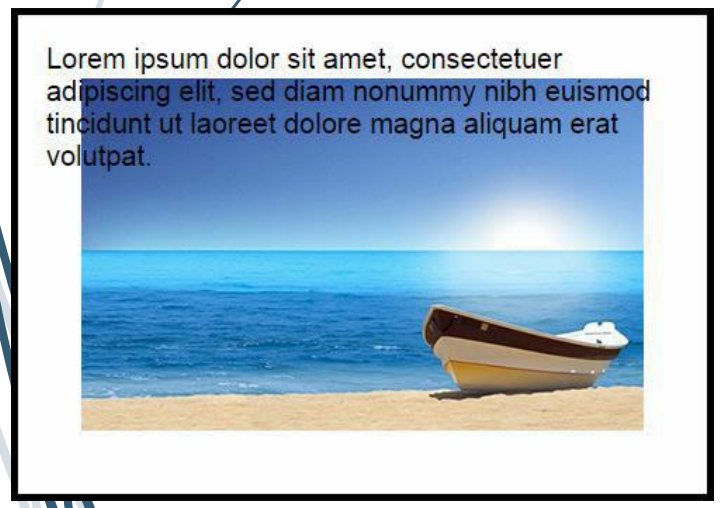
Il background

background-size è una proprietà che può essere utilizzata solo in presenza di background-image, e serve a fare in modo che l'immagine di sfondo si adegui alla grandezza del proprio contenuto.

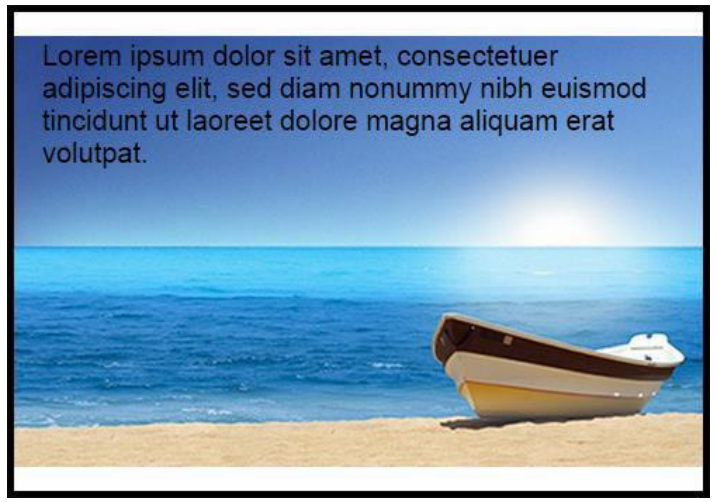
Accetta 3 valori (parole chiave):

- auto - l'immagine di sfondo copre l'area per quanto è in grado di fare con la propria superficie (valore di default)
- contain - l'immagine di sfondo copre l'area adeguandosi, in altezza o larghezza, al proprio contenitore
- cover - l'immagine di sfondo si adatta al contenitore a tal punto da coprirlo interamente, senza però perdere le proprie proporzioni

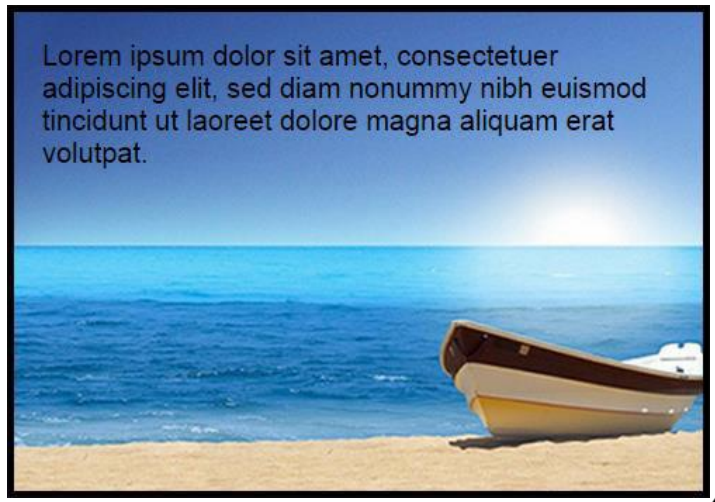
```
.box{background-size: auto;}
```




```
.box{background-size: contain;}
```



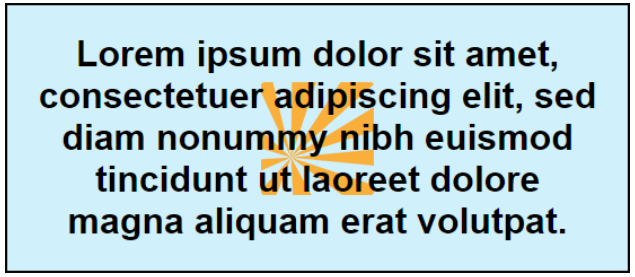

```
.box{background-size: cover;}
```



Il background

La proprietà **background** consente di scrivere più informazioni in un'unica istruzione.
In questo modo è possibile risparmiare alcune righe di codice e ottenere lo stesso risultato.

Per rappresentare il box qui accanto



potremmo scrivere il codice in due modi alternativi, ottenendo sempre lo stesso risultato:

```
h1 {
    width:500px;
    padding: 25px;
    border: solid 2px #000000;
    background-color: #d1f1fc;
    background-image: url(image.jpg);
    background-repeat: no-repeat;
    background-position: center center;
}
```

```
h1 {
    width:500px;
    padding: 25px;
    border: solid 2px #000000;
    background: url(image.jpg) no-repeat center center #d1f1fc;
}
```

In questo secondo caso alla proprietà background vengono associate (*in un solo colpo*) tutte le informazioni necessarie a indicare **l'immagine** e il **colore di sfondo**, il **posizionamento** dell'immagine e il **valore di repeat**.

La proprietà *visibility*

La proprietà **visibility** consente di nascondere un elemento della pagina.

Si differenzia dalla proprietà display: none, in quanto a un oggetto che abbia la proprietà visibility con valore "hidden" viene riservato comunque lo spazio fisico sulla pagina (come se fosse visibile) mentre un oggetto con proprietà display: none non occupa spazio sulla pagina finché tale proprietà non viene modificata.

Accetta 2 valori:

- visible - rende l'elemento visibile nella pagina (valore di default)
- hidden – nasconde l'elemento nella pagina, anche se comunque lo spazio ad esso riservato non può essere occupato da un altro elemento

```
p {
    width:500px;
    height: 250px;
    visibility: hidden:
}
```

In questo caso sarà costruito un elemento di blocco (<p>) che occuperà uno spazio a video di 500px di larghezza e di 250px di altezza, ma ne il box, nel il suo contenuto saranno visibili.

Le liste gestite dai CSS

Abbiamo visto che nel codice HTML è possibile ordinare parte del contenuto di una pagina utilizzando le liste ordinate (<ol>) o le liste non ordinate (<ul>).
La differenza tra le due liste è che nel primo caso accanto agli elementi della lista saranno visualizzati dei numeri, e nel secondo caso dei pallini.

Queste caratteristiche estetiche di base possono essere modificate attraverso i CSS.

Ci sono 4 proprietà su cui è possibile agire per modificare le etichette visibili in una lista:

- list-style-type -- determina il tipo di etichetta (pallino, quadrato, numero intero, numero romano, …)
- list-style-image -- permette di mostrare un'immagine come etichetta dei singoli elementi della lista
- list-style-position -- permette di settare l'etichetta interna o esterna al testo
- list-style -- è la proprietà che racchiude le 3 precedenti (di fatto le più utilizzata)

Le liste gestite dai CSS

Si può specificare lo stile dell'etichetta attraverso la proprietà **list-style-type** che accetta come valore una parola chiave.
Ci sono 9 parole chiave divise in gruppi:

- simboli:
  - disc (pallino) – etichetta di default per le liste
  - circle (cerchietto)
  - square (quadratino)

- numeri:
  - decimal (numeri decimali - 1, 2, 3, …) – etichetta di default per le liste
  - lower-roman (numeri romani minuscoli - i, ii, iii, ….)
  - upper-roman (numeri romani maiuscoli - I, II, III, ….)
  - lower-alpha (lettere minuscole a, b, c, ….)
  - upper-alpha (lettere maiuscole A, B, C, ….)
  - none (nessuna etichetta):

Per settare la proprietà **list-style-image** si dovrà utilizzare la parola chiave url(*percorso_immagine*).

Nel momento in cui si definisce una immagine come etichetta di un elenco, è consigliabile prevedere anche un simbolo (es. circle) che possa essere visualizzato nel momento in cui ci sia impossibilità di caricare l'immagine.

Le liste gestite dai CSS

Attraverso la proprietà **list-style-position** è possibile definire la posizione dell'etichetta di ciascun elemento dell'elenco:

- inside - all'interno del blocco di testo;
- outside - all'esterno del blocco di testo (valore di default).

Il valore "inside" fa sì che l'elenco risulti come un blocco di testo più compatto, in quanto l'etichetta viene posta all'interno del blocco, allineata con la prima riga di testo.

Utilizzando il valore "outside", invece, l'etichetta viene visualizzata all'esterno del blocco di testo, allineata con la prima riga.

Attenzione:
attraverso le proprietà degli elenchi è possibile definire il tipo di etichetta, nonché la posizione (interna o esterna), ma **non è possibile cambiare la distanza tra etichetta e testo**; al massimo si può agire sulla distanza tra il testo e il margine sinistro (margin-left oppure padding-left)

La proprietà *position*

La parte dei CSS relativa al posizionamento degli elementi è un argomento complesso, ma estremamente importante per le potenzialità che offre. Di fatto modificare il posizionamento di almeno un elemento è una prassi che si utilizza quasi in tutti i siti web.

position determina la modalità di presentazione di un elemento sulla pagina.
Serve, in pratica, a modificare il posizionamento di un elemento, togliendolo dal *normale flusso* della pagina, e dando ad esso delle coordinate ben precise che vanno di fatto a ignorare gli altri elementi presenti.

I valori con cui è possibile definire la modalità di posizionamento sono 4:

- static (valore di default)
- relative
- absolute
- Fixed

```
p { position: absolute }
.box{ position: fixed }
```

Fatta eccezione per la proprietà static, che non sortisce alcun effetto agli elementi della pagina, nel caso venga utilizzato uno degli altri valori sarà necessario aggiungere nel codice CSS anche altre proprietà, che servono a indicare le coordinate in cui fissare l'elemento.

Queste proprietà sono 4: left, right, top, bottom.
Possono essere *settate* con un valore assoluto (200px) o in percentuale (30%).

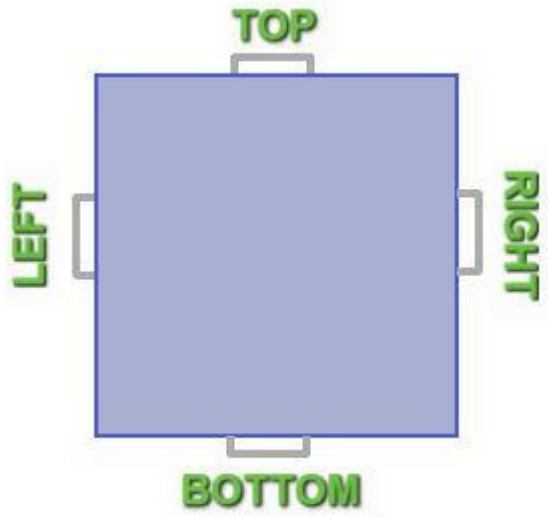
La proprietà *position*

position: relative
L'elemento viene posizionato relativamente al suo box contenitore.
In questo caso il box contenitore è rappresentato dal posto che l'elemento avrebbe occupato nel normale flusso del documento.
La posizione viene impostata con le proprietà top, left, bottom o right. Nel posizionamento relativo esse non indicano un punto preciso, ma l'ammontare dello spostamento in senso orizzontale e verticale rispetto al box contenitore.

In pratica, con il posizionamento relativo (position: relative), si va a modificare la posizione naturale di un elemento, traslandola attraverso l'uso di top, left, bottom o right. Un elemento posizionato relativamente non è rimosso dal flusso della pagina, ma solo spostato.

Ho scritto in precedenza che con top, left, bottom e right si riferiscono a un sistema di coordinate, ma in realtà si tratta di **vere e proprie traslazioni**.

La metafora più semplice per comprendere il funzionamento è immaginare che sui quattro lati di un elemento posizionato relativamente ci siano quattro maniglie che si possono tirare o spingere: **un valore positivo equivale a spingere**, mentre **un valore negativo equivale a tirare** l'elemento.



Per esempio, se viene specificato left: 30px significa che l'elemento viene spinto da sinistra di 30 pixel. Dichiarando invece top: -20px è come se tirassimo l'elemento dal suo lato superiore, con il conseguente effetto di traslarlo verso l'alto di 20 pixel.

Di fatto vanno specificati al massimo uno o due valori tra top, left, bottom e right.

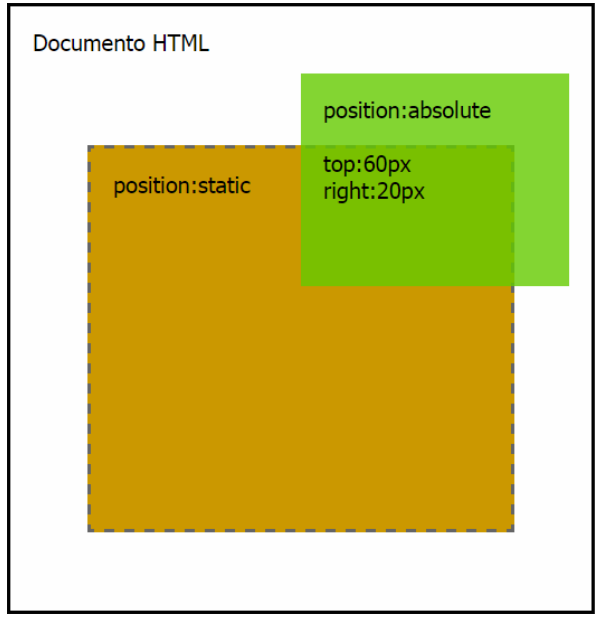
La proprietà *position*

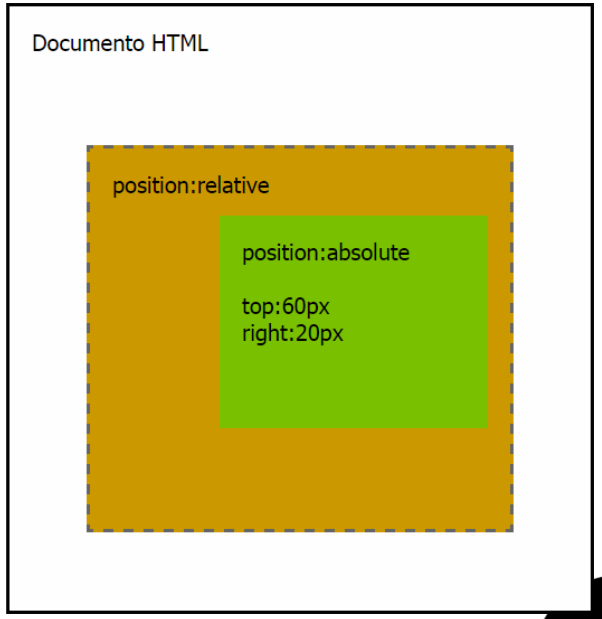
position: absolute
Con position: absolute l'elemento viene rimosso dal flusso del documento e posizionato in base ai valori forniti con le proprietà top, left, bottom o right.

Il posizionamento assoluto avviene sempre rispetto al box contenitore dell'elemento.
Questo è rappresentato dal primo elemento antenato (*ancestor*) **che abbia un posizionamento diverso da static**.

Se tale elemento non esiste, **il posizionamento assoluto avviene in base all'elemento radice html**, che in condizioni standard coincide con l'area del browser che contiene il documento e che ha inizio dall'angolo superiore sinistro di tale area.
Un elemento posizionato in modo assoluto scorre insieme al resto del documento.

Il modo più semplice per posizionare assolutamente un elemento rispetto ad un altro, consiste nel dichiarare, per quest'ultimo, **un posizionamento relativo senza coordinate**.





La proprietà *position*

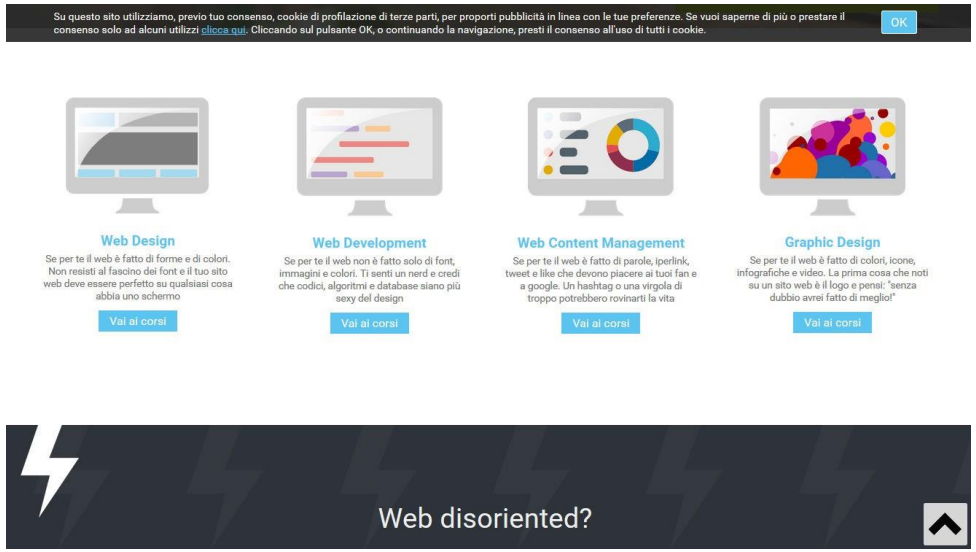
Un elemento con position: absolute è di default reso block-level, indipendentemente dalla sua natura iniziale. È come se, insieme a position: absolute dichiarassimo ogni volta implicitamente display: block.

Il fatto che tutti gli elementi posizionati assolutamente sono promossi a elementi block-level significa, in sostanza, che è possibile attribuire loro dimensioni esplicite. Non solo è possibile, ma è consigliabile impostare quantomeno una larghezza.

position: fixed
Usando questo valore, il box dell'elemento viene, come per absolute, sottratto al normale flusso del documento. La differenza sta nel fatto che per fixed il box contenitore è sempre la cosiddetta *viewport*. Con questo termine si intende la finestra principale del browser, ovvero l'area del contenuto.

Altra differenza fondamentale: **un box posizionato con fixed non scorre con il resto del documento**.
Rimane, appunto, fisso al suo posto.

Esempi di posizionamento fisso possono essere i box per le *cookie law* o le frecce (che appaiono solitamente in basso a destra) che servono a tornare su.



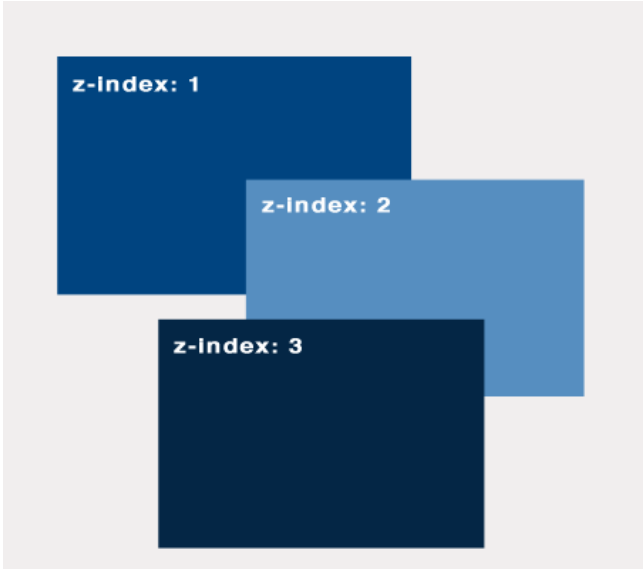
La proprietà *z-index*

La proprietà **z-index** imposta l'ordine di posizionamento dei vari elementi sulla base di una scala di livelli (verticali).
È un meccanismo simile a quello dei livelli sovrapposti di Photoshop.

I posizionamenti (assoluti, relativi o fissi che siano) permettono di sistemare o traslare elementi lungo due dimensioni (verticale e orizzontale).
C'è in realtà, nei CSS, una sorta di profondità, o terza dimensione: lo z-index, appunto.
Stabilisce la disposizione degli elementi posizionati lungo l'asse perpendicolare allo schermo.

z-index accetta, come valore, un numero intero.
Più alto è il numero, più alta sarà la posizione dell'elemento.
E' possibile associare a z-index anche numeri negativi.

```
.box1{ z-index: 1 }
.box2{ z-index: 2 }
.box3{ z-index: 3 }
```



Le pseudo-classi

Una pseudo-classe non definisce la presentazione di un elemento, ma di un particolare stato di quest'ultimo. In buona sostanza, grazie alle pseudo-classi possiamo impostare uno stile per un elemento al **verificarsi di certe condizioni**.
I nomi delle pseudo-classi, innanzitutto, sono preceduti dai due punti (:).
Una pseudo-classe segue, senza spazi, il nome del selettore e può essere associata a tutti i tipi di selettore.

```
selettore:pseudo-classe{ istruzioni di stile }
```

Alcuni esempi di utilizzo delle pseudo-classi:

- a:link {color: white;} -- pseudo-classe con elemento semplice
- p a:link {color: white;} -- pseudo-classe con selettore discendente
- a.classe:hover {color: white;} -- pseudo-classe con selettore di classe
- #id:hover {color: white;} -- pseudo-classe con selettore di id
- input[type=text]:focus {background: yellow;} -- pseudo-classe con selettore di attributo

Le pseudo-classi

La pseudo-classe **:link** definisce l'aspetto di partenza dei link, ovvero quello che ci troviamo davanti quando non sono stati visitati. Grazie ad essa possiamo personalizzare il colore, la decorazione del testo, la consistenza del font e altre proprietà degli elementi a superando le impostazioni di default dei browser, che di norma rendono i link con il colore blue e la sottolineatura.

```
a:link{ color: green }
```

La pseudo-classe **:visited** serve a modificare, se lo vogliamo, l'aspetto dei link quando siano stati visitati

```
a:visited{ color: red }
```

La pseudo-classe **:active**, come suggerisce il nome, serve a impostare la presentazione di un elemento quando e mentre esso viene attivato dall'utente. Tipicamente, interagendo con il mouse, un elemento è in stato :active mentre si tiene premuto il pulsante su di esso, ovvero per tutto il tempo che intercorre tra il click e il rilascio del pulsante.

```
a:active{ color: blue }
```

Le pseudo-classi

La pseudo-classe **:hover** (definita meglio come *pseudo-classe dinamica*) viene applicata quando si passa con il cursore del mouse (o altro dispositivo di puntamento) su un elemento senza attivarlo, ovvero senza cliccarci sopra. La sua applicazione più comune è quella per cui possiamo modificare l'aspetto di link e pulsanti al passaggio del mouse, ma viene usata anche per emulare nei CSS interazioni tipiche di Javascript, per esempio nella creazione di menu a tendina in cui una sezione compare e scompare, appunto, al passaggio del mouse.

```
a{
    text-decoration: none;
    color: red;}

a:hover{
    text-decoration: underline}
```

- In questo esempio i link sono tutti di colore rosso e non presentano il testo sottolineato.
- Al passaggio del mouse (con la pseudo-classe :hover) viene aggiunto, al codice già scritto, il codice CSS presente nella seconda della pseudo-classe :hover, con il risultato a video che i link mostrano il testo sottolineato.
- Non c'è bisogno, nelle pseudo-classi di modificare o ripetere tutti i parametri già impostati.

Una nota molto importante da fare è quella che le pseudo-classi **agiscono sull'elemento a cui sono associate**, ma **possono agire anche sugli elementi figli**.

```
li:hover a{text-decoration: underline}
```

In questo esempio, al passaggio con il mouse, non verrà modificata alcuna proprietà del TAG <li>, ma verranno modificati gli eventuali TAG <a> in esso contenuto, a prescindere che siano *figli diretti* o no del TAG <li>.

Pseudo-elementi

Queste particolari istruzioni CSS si definiscono **pseudo-elementi** perché è come se nel documento venissero inseriti nuovi elementi, che hanno la caratteristica di essere *fittizi*. In pratica sono presenti nel CSS ma non nel codice della pagina.

Il primo pseudo-elemento è **:first-letter.** Con esso è possibile formattare la prima lettera di qualunque elemento contenente del testo. Le proprietà modificabili sono ovviamente tutte quelle relative al carattere e al testo, ma anche quelle legate al colore, allo sfondo, ai margini, ai bordi e al padding.

```
p:first-letter {font-weight: bold}
```

Con lo pseudo-elemento **:first-line** è possibile formattare la prima riga di qualunque elemento contenente del testo.
Le proprietà modificabili sono ovviamente tutte quelle relative al carattere e al testo, ma anche quelle legate al colore, allo sfondo, ai margini, ai bordi e al padding.

```
p:first-line {font-weight: bold}
```

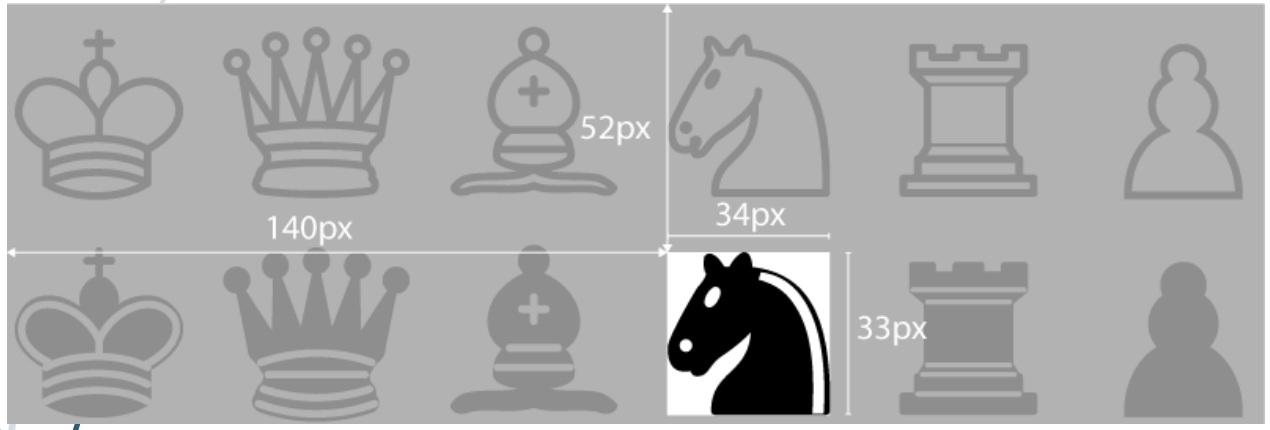
Altri due pseudo-elementi sono **:before** e **:after**.
Grazie a questi pseudo-elementi è possibile inserire nel documento HTML un contenuto non presente nel documento stesso.
Più precisamente, con :before si inserisce contenuto prima dell'elemento definito nel selettore, con :after dopo.
Il contenuto inserito con i due pseudo-elementi viene definito contenuto generato.

```
h3:before {content: " Io adoro i CSS3 ";}
a:after {content: url(immagine.png);}
```

Gli sprites CSS

Gli sprites CSS sono un modo per combinare insieme delle immagini al fine di ottimizzare il caricamento della pagina riducendo il numero di richieste HTTP al server.

L'idea, quindi, è quella di caricare una sola immagine composta da più immagini, e fare in modo che a video se ne veda solo una parte.

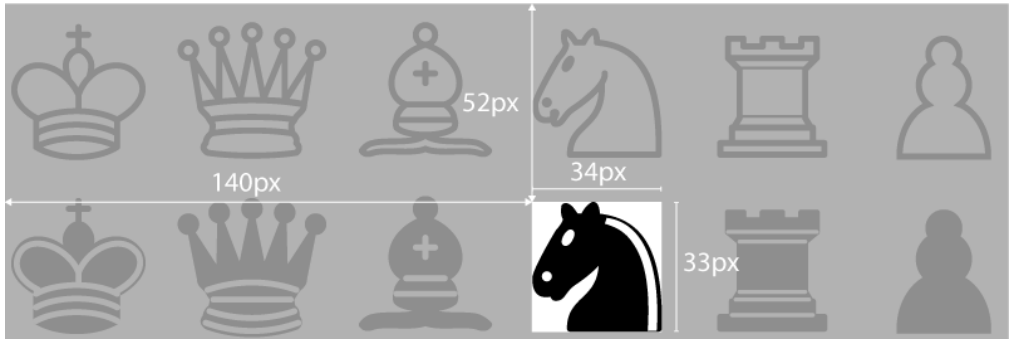


Le gestione degli sprites CSS è possibile grazie a 3 proprietà CSS che conosciamo benissimo:
width, height, background-position

Gli sprites CSS

Il primo passaggio da fare è quello di caricare l'immagine come sfondo di un TAG contenitore

```
.box{background: url(immagini/sprite.png);}
```



Per fare in modo di raffigurare a video l'immagine del cavallo (cosi come selezionato nell'immagine accanto) dovrò indicare al Browser di prendere in considerazione solo una piccola parte di tutta l'immagine.

Così come indicato nella figura, le dimensioni del «cavallo» sono 34px in larghezza e 33px in altezza.
Per cui il codice sopra scritto verrà modificato in questo modo

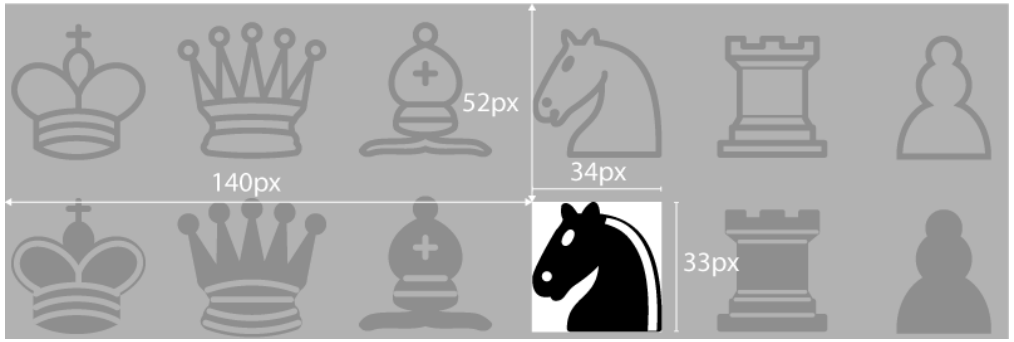
```
.box{
        background: url(immagini/sprite.png);
        width:34px;
        height:33px
}
```

Con il codice così scritto verrebbe mostrata l'immagine della corona in alto a sinistra, perché, senza indicare un posizionamento del background, il browser automaticamente da le coordinate top:0, left:0

Gli sprites CSS

L'ultimo passaggio, quindi, è quello di indicare le coordinate del background-position, in modo tale che l'immagine *scivoli* verso sinistra e verso il basso, in modo da porsi perfettamente sull'immagine del cavallo.

Per fare ciò dovrò lavorare sul posizionamento del background, così come nel codice che segue




```
.box{
	background: url(immagini/sprite.png);
	width:34px;
	height:33px;
	background-position:-140px -52px;
}
```

I valori sono espressi in negativo perché devo spostare l'immagine di sfondo verso sinistra di 140px e verso il basso di 52px;

L'uso degli sprites è molto raccomandato in situazioni come questa perché consente al Browser di caricare un'unica immagine (una sola risorsa esterna), ma di poterla frammentare in tante immagini più piccole.
Molto diffuso è l'uso dello sprite anche con la pseudo-classe hover.