

Copyright (c) 2008, 2009, 2010 Fabio Proietti

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Authors and contributors
Fabio Proietti

Feedback
Please direct any comments or suggestions about this document to
fabio.proietti AT istruzione DOT it

Publication date and version
2008-11-15, v.0.4
2009-11-15, v.0.5
2010-04-21, v.0.6

last modified 2013-02-01

Il flusso del testo

Il flusso normale del testo scorre normalmente nella pagina web da sinistra verso destra e dall'alto verso il basso. Esistono due tipi di elementi:

- gli elementi "inline" (ad esempio <a> e) che si affiancano a sinistra e vanno a capo solo quando è necessario, cioè quando incontrano il margine destro della finestra del browser.
- gli elementi "block" (come <p> e <h1>) che invece interrompono il flusso del testo e lo costringono ad andare sempre a capo.

Per esercizio, provare ad applicare le seguenti proprietà, prima ad un elemento block, come <p>, e poi ad un elemento inline, come <a>: text-align, border-style, border-color, margin, padding, width, height,...

Scoprirete che mentre su <p> funzionano tutte le proprietà, su <a> alcune funzionano e altre non funzionano...

Il selettore id

Come si possono fare due titoli usando due stili diversi?

Una possibilità è quella di usare gli stili inline.

Un'altra possibilità è quella di identificare in modo diverso ogni titolo. Ad esempio, per creare un identificatore si può scrivere:

```
<h1> primo paragrafo</h1>
<h1 id="speciale"> secondo paragrafo</h1>
```

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

Per utilizzarli nel codice HTML si aggiunge l'attributo id="..." mentre nel codice CSS si aggiunge il simbolo cancelletto (#) seguito dall'identificatore.

NOTA: Ogni identificatore può essere utilizzato per identificare un SOLO elemento <h1> in tutta la pagina HTML. Non posso riutilizzare la stessa regola per un altro titolo...

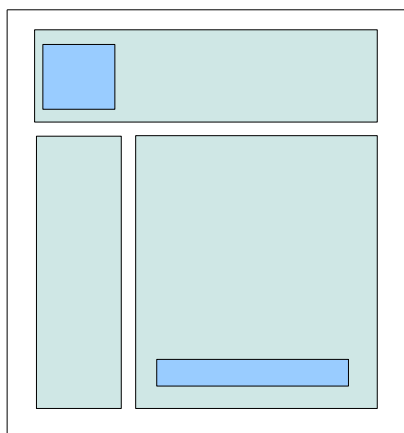
Immaginando di applicare un titolo con identificatore "speciale" in tutte le pagine di un sito web.

Se al posto dell'identificatore "speciale" si fosse usata una regola inline, come nel seguente esempio, si avrebbe avuto uno svantaggio: perché, quando si vuole cambiare il colore da rosso a verde, si deve modificare il codice html di tutte le pagine del sito.

```
<h1> primo paragrafo</h1>
<h1 style="color:red;"> secondo paragrafo</h1>
```

Elementi `` e `<div>`

Per organizzare il contenuto di una pagina, spesso si realizzano dei contenitori, come in questa figura:



Se si vuole attribuire una proprietà CSS ad un gruppo di oggetti o ad un testo, basta applicarla all'elemento HTML che li racchiude.

Ad esempio, se il testo è **racchiuso** da `<h1>`,

```
<h1>Esempio</h1>
```

basta creare una regola per il **selettore** `h1`.

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

Viceversa, se qualcosa non è racchiuso da un elemento HTML, non vi si può applicare nessuna proprietà CSS.

Per questo motivo, a volte è necessario introdurre nuovi elementi HTML per **racchiudere** qualcosa che non è ancora racchiuso da nessuno elemento HTML.

Questi nuovi tag HTML sono `` e `<div>`. Non hanno nessuna altra funzione se non quella di racchiudere. Per fare un esempio, anche `<h1>` è un elemento che potrebbe essere usato per racchiudere qualcosa, ma, oltre a questo, `<h1>` attribuisce anche un significato di titolo a tutto ciò che esso racchiude.

`<div>` e `` non attribuiscono nessun nuovo significato a ciò che racchiudono, vengono usati solo come dei contenitori neutri per poter creare nuove regole CSS.

`` è un elemento di tipo inline, viene usato per racchiudere solo qualche parola, come si potrebbe fare con `<a>`

`<div>` è un elemento di tipo block, viene usato per racchiudere molte cose, come si potrebbe fare con `<p>`.

Ad esempio, con `` si può racchiudere una parola di un paragrafo:

```
<p>La mia classe mi piace <span>molto</span> </p>
```

Ovviamente, per poter usare diversi stili per `` è necessario usare il selettore `class` (o il selettore `id`):

```
<p>La mia classe mi piace <span class="grassetto">molto</span> </p>
```

Il selettore class

L'identificatore (id) assicura che ci sia un solo elemento con tale nome, ma alcune volte si vuole riutilizzare la stessa regola per più elementi. Come si può ripetere la stessa regola su più elementi?

Una possibilità sarebbe quella di usare gli stili inline, con tutti i suoi svantaggi.

Un'altra possibilità è quella di usare due classi diverse per i paragrafi.

Ad esempio, per creare due classi si può scrivere:

```
<p class="blu"> primo paragrafo</p>
<p class="verde"> secondo paragrafo</p>
<p class="blu"> terzo paragrafo</p>
<p class="verde"> quarto paragrafo</p>
```

```
p.blu
{
    color: blue;
}

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

Per utilizzarle nel codice HTML si usa l'attributo class="..." mentre in CSS si usa il punto (.) seguito dal nome della classe.

NOTA: a differenza degli identificatori, una classe può essere utilizzata anche più volte dentro la stessa pagina web.

È inoltre possibile definire contemporaneamente più classi sullo stesso elemento, ma con proprietà diverse (come colore ed allineamento) e combinarle insieme.

```
<p class="blu destra"> la prima frase</p>
<p class="blu sinistra"> la seconda frase</p>
<p class="rosso destra"> la terza frase </p>
<p class="rosso sinistra"> la quarta frase </p>
```

```
p.blu
{
    color: blue;
}

p.rosso
{
    color: red;
}

p.destra
{
    text-align:right;
}

p.sinistra
{
    text-align:left;
}
```

I collegamenti e le pseudoclassi

I collegamenti (o link) possono avere le stesse proprietà viste anche per il testo, ad eccezione delle proprietà per elementi di tipo block. Esistono però altri tipi di selettori (che si chiamano anche pseudoclassi):

`a:link` collegamento non visitato

`a:visited` collegamento visitato

`a:hover` collegamento sotto al puntatore del mouse

`a:active` collegamento mentre viene premuto il tasto per selezionarlo

in questo esempio i link vengono colorati di rosso e cambiano colore quando si avvicina il puntatore del mouse

NOTA: è importante specificare le quattro pseudoclassi sempre nell'ordine indicato

```
a:link {
    color: #ff0000;
}

a:hover {
    background-color: #00ff00;
}
```

Una pseudoclasse con classe

Nel caso in cui si voglia usare una classe con le pseudoclassi dei collegamenti, si deve indicare prima la classe e poi la pseudoclasse

```
a.interno:link
{
    color: #ff0000;
}

a.esterno:link
{
    color: #00ff00;
}
```

Esercizio

creare una tabella come quelle indicate nelle seguenti figure, utilizzando le classi in css in fogli di stile esterni

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

soluzione...

prima	seconda	terza	quarta
a	b	c	d
a	b	c	d
a	b	c	d
a	b	c	d

soluzione...

Alterare il flusso del testo


La proprietà `display` consente di modificare questa caratteristica.

Esercizio

Provare ad utilizzare su due immagini `display:inline`, `display:block` e `display:none`

Proprietà	Valori permessi	
display	none block inline	table table-row inherit
visibility	visibile	hidden

Insieme a `display`, esiste una proprietà leggermente diversa: `visibility`

parola con  immagine	<code>display: inline;</code> <code>visibility: hidden;</code>	valori predefiniti per le immagini
parola con immagine	<code>visibility: hidden;</code>	rende trasparente (cioè nascosto o invisibile) l'elemento a cui è applicata.
parola con immagine	<code>display: none;</code>	rimuove completamente dalla pagina web l'elemento a cui viene applicata.

Le proprietà "`position`" e "`float`" permettono di modificare il flusso del testo, modificando il metodo di posizionamento di un elemento, rispetto al bordo della pagina o rispetto ad un elemento HTML padre (contenitore).

Proprietà	Valori permessi	
position	<i>static</i> <i>absolute</i>	<i>fixed</i> <i>relative</i> <i>inherit</i>
float	left right	none inherit

`Position` non indica "dove" mettere l'elemento, ma solo il sistema da adottare. La differenza tra "`position`" e "`float`" si nota quando due utenti osservano la stessa pagina utilizzando una finestra del browser con dimensioni diverse oppure modificando la dimensione della finestra del browser.

- Utilizzando la proprietà "`position`" l'elemento può essere messo ovunque, anche sopra altri elementi, addirittura nascondendoli.
- Usando la proprietà "`float`" gli altri oggetti possono scorrere liberamente a destra (o a sinistra). Ad esempio, la proprietà "`float`" permette al testo di scorrere ordinatamente intorno ad un'immagine.

Le due proprietà potrebbero essere usate anche contemporaneamente, ma nel caso in cui venga specificata la proprietà `position`: `absolute` qualsiasi valore assegnato a `float` verrà ignorato.

Proprietà `position`

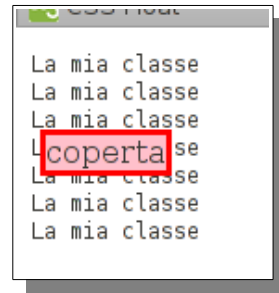
Prima di vedere i valori che può assumere la proprietà `position`, è necessario conoscere altre quattro proprietà che si usano INSIEME a `position`. Infatti, dopo aver specificato il valore di `position`, si deve specificare il valore della distanza dai margini della pagina web. Se si indica la distanza dal margine superiore, ovviamente, non si può specificare anche quella dal margine inferiore: `top` è alternativa a `bottom`, e `right` è alternativa a `left`.

Proprietà	Valori permessi	
top / left / right / bottom	<i>auto</i> <i>inherit</i>	VALORE IN PIXEL VALORE IN %

Valori <code>position</code>	Descrizione
<code>static</code>	Default. Non specifica la posizione e l'elemento appare nel normale flusso.
<code>absolute</code>	Fissa la posizione <u>rispetto</u> al contenitore superiore (padre) che <u>non sia static</u> . Il valore deve essere specificato nelle proprietà: " <code>left</code> ", " <code>top</code> ", " <code>right</code> " e " <code>bottom</code> ".
<code>fixed</code>	Fissa la posizione rispetto alla pagina web. Il valore è specificato come sopra in " <code>left</code> ", " <code>top</code> ", ecc.
<code>relative</code>	Fissa la posizione <u>rispetto</u> alla posizione normale (<code>static</code>) dell'elemento. Il valore è specificato come sopra.
<code>inherit</code>	Eredita la proprietà <code>position</code> dall'elemento contenitore (padre)

Nel seguente esempio si usa il valore "fixed" per <p> per poterlo poi disporre vicino alla parte bassa rispetto all'elemento <body>, nel seguente modo:

```
p {
  position: fixed;
  bottom: 50px
  right: 50px
  margin: 5px;
  border: 3px solid red;
  background-color: pink;
}
```



Esercizio

inserire un box con bordo rosso sulla parte destra in basso della pagina e al suo interno un testo che sia allineato a destra.

Proprietà float

Un elemento "float" si posiziona adiacente agli elementi successivi, ma non "sopra" come accade usando la proprietà "position"

NOTA: un elemento "float" non influenza gli elementi precedenti

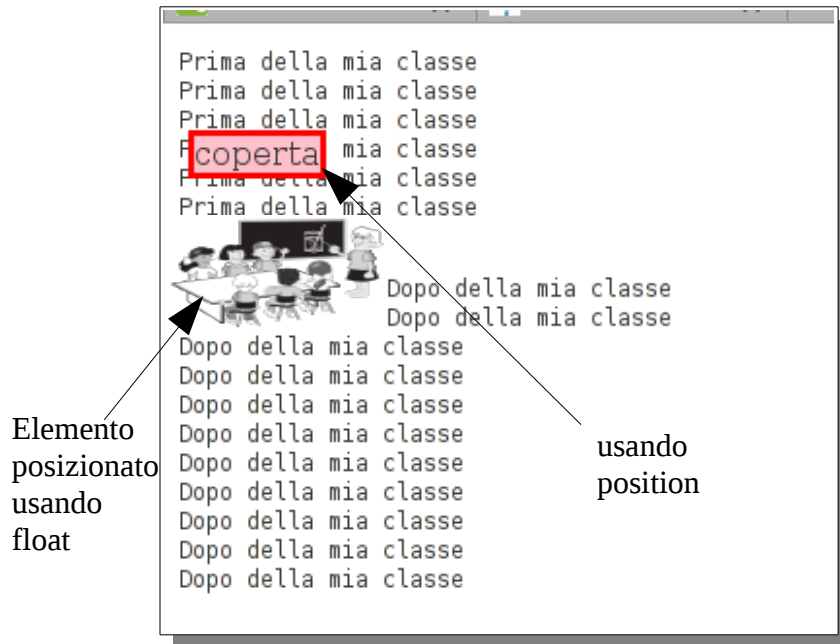
Valori float	Descrizione
left	l'elemento si posiziona a sinistra e tutto quello che lo segue gli scorre intorno (a destra)
right	l'elemento si posiziona a destra e tutto quello che lo segue gli scorre intorno (a sinistra)

Per il prossimo esempio è stato usato il seguente codice

```
<p> Prima della mia classe </p>
<p> Prima della mia classe </p>

<p> Dopo la mia classe </p>
<p> Dopo la mia classe </p>
```

```
img {
  float:left
}
```



La precedente regola posiziona l'elemento (un'immagine) allineandolo completamente a sinistra. Come sarebbe stato il risultato senza usare float? Di solito gli altri elementi scorrono dalla parte opposta (a destra).

Per "interrompere" il flusso del testo a fianco di un oggetto float, come il precedente, si può applicare su alcuni elementi, la regola `clear:left;` (o `clear:right;`)

Un elemento che possiede la proprietà clear, si disporrà subito **sotto** un eventuale elemento float.

Sintassi dei selettori

In CSS, all'inizio di ogni regola può essere specificato l'elemento, o tag, a cui verrà applicata. Ad esempio, la seguente regola verrà applicata agli elementi `<p>`:

```
p {
  text-decoration:underline;
}
```

La seguente regola verrà applicata sia agli elementi `<h1>` che `<p>`

```
h1, p {
  text-decoration:underline;
}
```

La seguente regola verrà applicata solo agli elementi `` interni a `<p>` mentre non a quelli che sono interni ad altri elementi (come nella seconda riga di codice HTML, dove `` è interno all'elemento `<a>`)

```
p > img{  
    background-color:yellow;  
}
```

codice HTML <p> </p>

codice HTML <p> <a> </p>

La seguente regola verrà applicata a tutti gli elementi che si trovano all'interno di elementi <p>

```
p img{  
    background-color:yellow;  
}
```

codice HTML <p> </p>

codice HTML <p> <a> </p>

Esempio

Esempio che fa apparire e scomparire un'immagine nascosta

```
img.nascosta
```

```
{  
  display:none;  
}
```

```
a.sorpresa
```

```
{  
  border-style: solid;  
  width:20;  
  height:20;  
}
```

```
a.sorpresa:hover img.nascosta
```

```
{  
  display: inline  
}
```

Questo esempio permette di osservare una combinazione di tutte le proprietà CSS viste fino ad ora per creare un menu a tendina in css (senza usare programmi in javascript...)

```
ul.mainmenu, ul.mainmenu ul {  
    width: 85px;  
    border-width: 1px;  
    border-style: solid;  
    border-color: black;  
    background-color: lightyellow;  
}
```

```
ul.mainmenu li {  
    position: relative;  
}
```

```
ul.mainmenu li:hover {  
    background-color: yellow;  
}
```

```
ul.mainmenu li > ul {  
    display: none;  
}
```

```
ul.mainmenu li:hover > ul {  
    display: block;  
    /*  
    position: absolute;  
    top: 2px;  
    left: 80px;  
    */  
}
```

