Guida ai fogli di Stile CSS

I CSS, ovvero Cascading Style Sheets sono ormai molto comuni sul web, servono per migliorare l'aspetto estetico ed al tempo stesso facilitano la creazione e la manutenzione di siti web e questo a prescindere che si tratti di poche pagine o di grossi portali.

Se combinati con un linguaggio di scripting, quale per esempio JavaScript, danno vita al DHTML ovvero un HTML Dinamico, consentendo di superare quelli che erano considerati un tempo i limiti di html standard.

Con questa tecnica è possibile creare persino delle vere e proprie animazioni sfruttando l'elevata versatilità offerta dal posizionamento degli oggetti sullo schermo, siano essi grafici oppure no, posizionamento che a differenza del solo html è ora possibile fare con accurata precisione.

Come quasi tutte le cose di questo mondo vanno create, per cui scegliete cosa preferite fare, se procurarvi un editor specifico per CSS oppure adoperare lo stesso editor che usate abitualmente per le vostre pagine html.

Se non avete un editor specifico vi ricordo che anche in questo caso, così come per html, serve un editor puro (ASCII), il notepad o il wordpad a corredo di windows, per esempio, andranno benissimo. Il mio consiglio è quello di procurarvi un editor per CSS, vi sarà di maggiore aiuto. vi consiglio per esempio TopStyle nella sua versione lite free.

Vorrei ribadire che per fare uso di questa "tecnica" si deve necessariamente conoscere un minimo di html essendo di fatto i fogli di style un'estensione di html stesso. Senza le conoscenze di base di questo linguaggio sarebbe davvero difficile capirli ed utilizzarli al meglio.

Dovendo illustrarli in modo sintetico direi che non si usano solo per giustificare il testo o stabilirne i suoi rientri; pensate alle dimensioni dei font di caratteri, non esistono più limiti. E le immagini? queste possono essere sovrapposte come se si trattasse di piani trasparenti dove ad ognuno è possibile assegnare una diversa priorità di visualizzazione. In poche parole: se non ci fossero si dovrebbero inventare.

Da non trascurare neppure l'aspetto pratico legato al risparmio di tempo e di energie in fase di stesura e modifiche. Provate ad immaginare un sito composto da decine (a volte centinaia) di pagine e che queste facciano tutte riferimento ad uno o più fogli di stile esterni, sarà sufficiente una veloce modifica a quel foglio per vedere immediatamente la modifica applicata a tutte le pagine che lo richiamano.

C'è soltanto da chiedersi come mai un sistema che a quanto pare offre solo vantaggi abbia trovato difficoltà nel raggiungere il successo che attualmente merita.

La sola possibile spiegazione che ho trovato è stata quella che inizialmente non tutti i browser li supportavano ed ancora oggi non tutti li supportano allo stesso modo. Fra i vari browser ci sono differenze a volte davvero sorprendenti e tali da scoraggiare chi fosse indeciso nel farne uso.

Attualmente le cose sono decisamente cambiate a favore dei CSS, forse perché è venuto meno il predominio del browser IE, era lui infatti che più di altri non rispettava le regole dettate dal W3C (organo ufficiale che si preoccupa di stabilirne le regole). Di fatto trovare ad oggi un sito che non ne faccia uso è cosa davvero rara.

Finita questa premessa credo si possa passare al lato pratico per vedere finalmente come si adoperano.

Su questa guida per facilitarne la comprensione ho inserito in box a sfondo verde tutto quello che riguarda il codce CSS mentre nei box a sfondo giallo tutto quello che riguarda il codice html visto che le due cose si integrano a vicenda.

```
codice CSS
```

```
codice HTML
```

```
risultato degli esempi
```

Per prima cosa devono essere racchiusi in un elemento (tag) che viene specificato in html, si tratta dell'elemento: <style> e relativa chiusura </style>. Questo elemento serve ad informare il browser che si tratta di stili e che questi apporteranno le dovute modifiche ai vari elementi (tags) di HTML definiti al loro interno, per tutti i tags, nessuno escluso, dipenderà soltanto dalle combinazioni che si vorranno creare.

Fondamentalmente sono tre i modi di usare i css, quale sia il migliore dipende soltanto da voi, dalla vostra organizzazione mentale e dalle vostre reali esigenze:

1. Direttamente in linea
2. Ad inizio pagina a stile incorporato
3. Nel collegamento ad un foglio di stile esterno

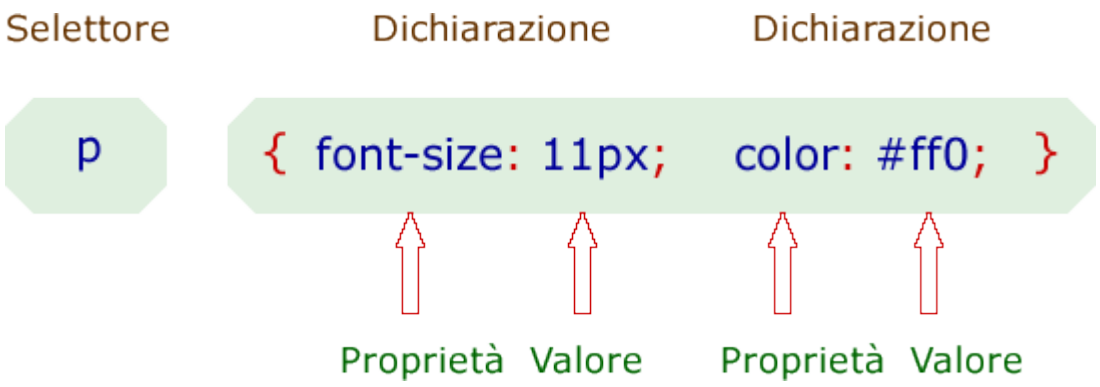
La sintassi Abbiamo detto che hanno un loro elemento (tag) che viene specificato in html, si tratta dell'elemento: <style> con relativa chiusura </style>. Questo elemento serve per informare il browser che si tratta di stili che apporteranno le dovute modifiche agli elementi di html in esso definiti. La dichiarazione di style deve essere collocata all'interno dei tags <head> e </head> della pagina web, unica eccezione per le definizioni direttamente in linea dove basta il solo style come attributo del tag utilizzato (lo vedremo più avanti). E' possibile specificare il tipo di style, di solito text/css ed il media, cioè a quale dispositivo si riferisce, di solito screen o print rispettivamente per schermo o stampante. Altri possibili media sono: all, braille, handheld, tv.

Abbiamo detto che fondamentalmente sono tre i modi di usare i css, quale sia il migliore lo lascio decidere a voi, sappiate che l'uso di un modo non esclude gli altri due, al contrario, si combinano alla perfezione, dipende esclusivamente da voi stessi e dal vostro modo di organizzarvi, oltre che dalle reali esigenze:

1. Direttamente In linea
2. Ad inizio pagina a stile incorporato
3. Nel collegamento ad un foglio di stile esterno

Ognuno di questi tre possibili modi differisce leggermente dall'altro, lo vedremo più avanti quando li analizzeremo uno per uno nel dettaglio.

Le regole comuni a tutti e tre i possibili modi sono quelle che gli attributi devono essere inseriti all'interno di parentesi graffe { } laddove in HTML verrebbe usato un "=" (uguale) viene invece usato ":" (due punti). Argomenti consecutivi sono separati da un ";" (punto e virgola) invece che dalla "," (virgola). Inoltre molti argomenti sono divisi da "-" (trattino centrale) che fa parte integrante del nome stesso.



Gli spazi vuoti ed i ritorno a capo non hanno alcuna influenza.

Vediamo un possibile style. Immaginiamo di volere assegnare al nostro testo una leggera indentazione (12 pixel) con allineamento giustificato che tradotto significa: avere un leggero rientro del testo ad inizio paragrafo con allineamento tabulato su entrambi i lati, esattamente come quello che state vedendo adesso in questo paragrafo.

```
<head>

<style type="text/css">

p {text-align: justify; text-indent: 12px;}

<style>

</head>
```

Prima di tutto individuiamo nella nostra pagina html la zona head ed al suo interno creiamo una dichiarazione di foglio di style adoperando i tags style e /style come da esempio sopra. Quindi definiamo il tag <p> (paragrafo) di html associandogli l'attributo text-align con il parametro *justify* per il testo giustificato, mentre l'attributo text-indent con il parametro *12px* per l'indentazione di 12 pixel dal suo margine sinistro ad inizio paragrafo.

Da notare in questa fase di apprendimento soltanto la sintassi: Elemento, apri parentesi graffa, attributo, due punti, parametro, punto e virgola, attributo, due punti, parametro, punto e virgola, chiudi parentesi graffa.

Per conoscere quali sono gli attributi ed i relativi parametri possibili, vi rimando alle voci del menù laterale gli attributi di stile e i parametri dove è possibile vedere un'ampia panoramica, più o meno completa, di quelli che sono i più comuni ed i più adoperati.

Abbiamo detto che uno stile può essere associato ad un elemento valido di html, per cui se associamo il nostro stile all'elemento <body> sarà come aver definito uno style per tutta la pagina web dal momento che body in html è l'elemento che comprende tutto il corpo della pagina (nell'esempio precedente ci eravamo limitati al solo tag P di paragrafo).

Da notare che se un elemento è contenuto all'interno di un altro elemento questo ne eredita le sue proprietà, per cui un paragrafo essendo di fatto contenuto all'interno di body ne eredita le sue proprietà, a meno che queste non vengano cambiate ridefinendo il paragrafo stesso. Alcuni esempi di possibili stili:

```
<style type="text/css">

body {font-size: 10pt; color: Blue;}
strong {font-size: 12pt; color: Red;}
```

```
em {font-size: 9pt; font-style: italic; color: Green;}

</style>
```

Vediamo i tre esempi sopra, il primo: body ha un font formato da 10 punti (10pt) ed usa un colore blu scuro, sarebbe stato valido anche una specificazione del colore con notazione esadecimale: #000099 (che personalmente preferisco) corrisponde sempre al blu, è valida anche questa abbreviazione #009. Essendo assegnata al body, tutto il testo all'interno di quella pagina seguirà questo stile perchè l'elemento di html body si riferisce a tutto il documento, praticamente quello che ho fatto io con questa pagina.

strong (o bold in html) aumenta il font di 2 punti rispetto a quello definito in body così da evidenziarlo maggiormente ed in più ne cambia anche il colore passando dal blu al rosso.

em (o Italic in html) cambia lo stile in Italic e *riduce il font di un punto* (9pt) rispetto a quello standard definito nel body (10pt) tre rispetto al bold (12pt) di strong e gli cambia colore passando al verde (Green).

E' anche possibile associare le stesse definizioni a più elementi:

```
<style type="text/css">

h1, h2, p, div {font-size: 20px; color: #ff9900;}

</style>
```

In questo caso l'utilizzo di uno qualunque dei tag: <h1>, <h2>, e produrrà lo stesso identico effetto: font di dimensione 20 pixel di colore arancio.

Tag come ridefinito sopra

Bene, se avete capito il meccanismo, se vi risulta tutto quanto chiaro, non resta che passare alle varie tecniche per associare lo stile ai diversi elementi html e per fare questo vi ricordo che i modi possibili sono più di uno.

Iniziamo da quello più semplice ed immediato, lo stile in linea.

In linea quando si ha la necessità di applicare lo stile al solo blocco o parte di codice che stiamo trattando, senza cioè che questo stile vada ad influire su altre parti della stessa pagina web.

Per esempio un paragrafo che debba avere determinate caratteristiche ma con la certezza che queste *non* siano estese a tutti gli altri paragrafi contenuti all'interno della stessa pagina web.

Per riprendere lo stile dell'esempio precedente, abbiamo detto di volere una leggera indentazione con testo giustificato, volendo applicare questa scelta ad un solo paragrafo, il modo più comodo e rapido è quello di associare lo stile al solo elemento interessato.

Questo esempio pratico chiarirà sicuramente meglio:

```
<p style="text-align: justify; text-indent: 12px;">... </p>
```

Tutto ciò che sarà scritto tra l'elemento di apertura <p style="... >(paragrafo) e fino al suo elemento di chiusura avrà come stile: giustificato con una indentazione di 12 pixel;

Si possono combinare fra di loro più stili, la sintassi corretta prevede di racchiudere il tutto fra doppi apici separando i vari attributi di stile con il punto e virgola.

I più attenti si saranno accorti dell'importanza di fare uso dell'elemento (tag) di chiusura, in questo caso il che mentre per HTML non era obbligatorio, lo diventa invece nell'uso dei CSS. La chiusura di questo elemento determina infatti anche la chiusura (e quindi la fine) dello stile, per cui meglio prendere fin da subito la buona abitudine di chiudere sempre tutti gli elementi (tags) aperti.

Si noti come lo stile in linea sia stato associato all'elemento , semplicemente avendo introdotto l'attributo style all'interno delle stesse parentesi angolari <> senza bisogno di specificare type css/txt visto in precedenza.

Allo stesso identico modo avrebbe potuto essere associato a qualsiasi altro elemento valido di HTML (sono esclusi elementi come <br>) ne consegue una facile personalizzazione di tutti (o quasi) gli elementi di html, anche quelli che non piacciono molto o servono a poco e magari proprio per questo non vengono quasi mai adoperati.

Con i CSS si adoperano in modo particolare due elementi di html: div e span che altrimenti non avrebbero molta ragione per essere utilizzati visto che fanno solo da "*contenitore*", questi due elementi non producono alcun effetto se adoperati da soli ma ad essi è possibile associare tutti i vari styli. Vediamoli nel dettaglio, si tratta rispettivamente di:

 e

 si applica ad un blocco di codice e provoca un ritorno a capo con la sua chiusura mentre può essere applicato in qualsiasi punto del documento (anche all'interno di un) senza che questo ne alteri la struttura, praticamente é trasparente al resto del codice visto che non provoca alcun ritorno a capo limitandosi praticamente a cambiare soltanto la parte di testo ad esso associata.

Per il nostro esempio sopra, avrei potuto ottenere la stessa cosa usando un contenitore al posto del paragrafo in questo modo:

```
<div style="text-align: justify; text-indent: 12px;"> testo </div>
```

Tenete sempre bene in considerazione la differenza fra e , del fatto che div può essere usato come piccolo o grande contenitore, la sua chiusura provoca un ritorno a capo ed è quindi usato per interi blocchi di codice, siano essi paragrafi di testo o immagini da posizionare sullo schermo, mentre span si utilizza all'interno dei paragrafi stessi e magari per modificare soltanto parti di testo che lo compongono senza provocare alterazioni alla struttura come il ritorno a capo. Mettere span dentro a div è corretto, mettere div dentro span è sbagliatissimo.

Da notare infine la possibilità di nidificare fra loro più contenitori all'interno di altri elementi o di altri contenitori:

```
<div style="font-size: 10pt; color:#006600;">testo di prova con stile div

<p style="font-size: 12pt; text-align: center;">testo del paragrafo con stile div + stile p
testo allineato al centro 12 pt</p>

testo di prova stile div senza lo stile di p
</div>
```

Questo il risultato:

testo di prova stile div = 10pt colore verde

testo del paragrafo con stile div + stile p
testo allineato al centro 12 pt

testo di prova stile div senza lo stile di p

* * assegna a tutto il blocco di testo, fino alla chiusura * * un font di 10pt ed un colore verde, il * * assegna al paragrafo un font più grande: 12 pt ed un allineamento del testo centrato, la chiusura di * * mette fine al testo allineato al centro e alle dimensioni 12pt e tutto torna come definito dal div iniziale e resta valido fino alla sua chiusura * *

Volendo scrivere in rosso solo alcune parole senza alterare la struttura, si userà span che abbiamo detto è un contenitore in linea che non provoca alterazioni se non per quello che viene dichiarato al suo interno.

```
<div style="font-size: 10pt; color:#006600;"> testo di prova con stile div

<p style="font-size: 12pt; text-align: center;">testo del paragrafo con stile div + stile p testo allineato al centro 12 pt e <span style="color:#ff0000;">scritta in rosso</span></p>

testo di prova stile div senza gli stili di p e di span
</div>
```

Questo il risultato:

testo di prova stile div = 10pt colore verde

testo del paragrafo con stile div + stile p
testo allineato al centro 12 pt e scritta in rosso

testo di prova stile div senza gli stili di p e di span

* * assegna a tutto il blocco di testo, fino alla chiusura * * un font di 10pt ed un colore verde, il * * assegna al paragrafo un font più grande: 12 pt ed un allineamento del testo centrato, al suo interno con ** si definisce una parte di testo in rosso, la chiusura di ** mette fine al testo rosso, quella di * * mette fine allineato al centro con dimensioni 12pt e tutto torna come definito dal div precedente e resta valido fino alla sua chiusura * *

A style incorporato Cioè quando ad un determinato stile fanno riferimento più elementi (tags) html uguali nella stessa pagina. Sarebbe estremamente dispersivo, oltre che laborioso, definire sempre lo stesso stile più volte all'interno della stessa pagina visto che servirebbe sempre per lo stesso identico scopo. Molto meglio definirlo una sola volta, ad inizio pagina. Così facendo tutti gli elementi interessati di quella pagina seguiranno lo stesso stile.

In questo caso le istruzioni non dovranno più essere inserite all'interno di uno specifico elemento come abbiamo visto nella spiegazione in linea ma saranno dichiarate ad inizio pagina, usando il tag (tags) <style> e </style> posto all'interno della sezione <head> </head>

Questo metodo non impedisce di fare uso anche di eventuali definizioni di stile in linea, anzi, queste avrebbero priorità sulle dichiarazioni di stile inserite ad inizio pagina.

Da notare che in questo modo si renderanno necessarie le parentesi graffe per delimitare i vari blocchi di istruzioni relativi agli elementi interessati, cosa che non serviva per gli styli in linea.

Facendo riferimento ancora all'esempio precedente:

```
<style type="text/css">
<!--
p {
text-align: justify;
text-indent: 12px;
}
-->
</style>
```

Adesso tutto ciò che si troverà all'interno di qualsiasi paragrafo definito con la consueta forma html ... subirà le impostazioni dello stile dichiarato ad inizio pagina, a meno che non venga creato uno stile in linea per uno o più determinati paragrafi, i quali, come ho detto sopra, avrebbero priorità rispetto a quanto definito nello style ad inizio pagina.

Ripeto ancora una volta l'importanza della chiusura degli elementi aperti, anche nei casi in cui si trattasse di elementi come che in html non necessiterebbero della relativa chiusura Non chiudere un elemento significa non porre fine allo stile ad esso associato ma estenderlo a tutto il documento, cosa sbagliata e quasi mai voluta.

In questo contesto sono proprio loro: i tags chiusi, a decretare la fine dello style oltre a quella del paragrafo. Dimenticarli significa dire al browser di estendere i parametri dello style a tutto il resto del documento.

Spiegando il codice dell'esempio: i due elementi di commento <!-- e -->, posti ad inizio e fine blocco, sono gli stessi adoperati in html e servono per fare ignorare lo style a tutti quei browser obsoleti che non dovessero supportarlo. Da non usare mai per lasciare commenti all'interno della dichiarazione di stile, non è come in HTML, se serve commentare all'interno di un foglio di style si adoperano /* (barra e asterisco) per inizio commento e */ (asterisco barra) per la sua fine.

```
<style type="text/css">
<!--
p {
text-align: justify; /* commento per allineamento giustificato */
color: #ff0000; /* commento per colore rosso */
}
-->
</style>
```

La stesura del codice io l'ho scritta così, con i ritorni a capo, solo per dargli un certo ordine ma nessuno vieta che possa essere scritta tutta su una riga, l'importante è rispettare i simboli: parentesi, due punti, spazi, punti e virgola.

```
<style type="text/css">
<!--
p {text-align: justify; color: #ff0000;}
-->
</style>
```

Nell'esempio è stato adoperato l'elemento (tag) e senza parentesi angolari, nei fogli di style infatti si deve usare il tag o elemento ma non le sue parentesi angolari che invece fanno parte di html. Avrete sicuramente capito che al posto di p potevamo usare qualsiasi altro elemento valido di html: b, strong, H1, table, ecc. ecc.

Vista questa geniale possibilità, perché non ridefinire elementi di html che non si adoperano perchè non piacciono o personalizzare quelli più adoperati per rendere il tutto più gradevole?

Così facendo sarebbe come avere a disposizione dei nuovi comandi html, o meglio, dei nuovi elementi, con in più il fatto di averli personalizzati a proprio piacere, se non è versatilità questa.

Personalmente faccio poco uso di elementi ridefiniti, preferisco molto di più usare delle "classi" (le vedremo più avanti) che creo sul momento e alle quali associo nomi decisamente più significativi, per me, per meglio capire a cosa mi servono realmente. Non esistono limiti al numero di classi definibili dall'utente, per cui potrei avere un solo paragrafo con dieci classi diverse, evviva la fantasia di ognuno di noi.

Foglio di style esterno

Abbiamo visto come definire uno stile in linea quando ci serve magari una veloce particolarità delimitata ad una sola zona del nostro documento.

Abbiamo visto anche come definire uno stile ad inizio pagina al quale tutto il documento faceva riferimento ed uso.

Vediamo adesso quello che è in assoluto il più comodo e pratico dei sistemi, quello in cui esiste un foglio di stile esterno, separato dalla pagina web, che racchiude tutte le dichiarazioni di style, e che viene richiamato da tutte le pagine web che compongono il sito. In questo foglio, o fogli visto che possono essere più di uno solo, andremo a scrivere tutti gli stili che ci servono, comprese le "classi" e gli "id".

Anche nel caso di uno o più fogli di style esterni, sarà sempre possibile definire all'occorrenza stili ad inizio pagina e stili in linea, questi avrebbero priorità su quanto dichiarato nel foglio di style esterno.

Cosa significa foglio di stile esterno?

Significa che se gli stili da definire sono molti, se più di una sola pagina web ne farà uso, la cosa migliore da fare è sicuramente quella di creare una struttura esterna alla quale potranno accedere tutte le pagine interessate. Questa struttura altro non sarebbe che il foglio di style, un documento di normalissimo testo con estensione finale .css dove al suo interno saranno contenute tutte le dichiarazioni valide per un foglio di style. Valgono le stesse regole sintattiche viste fino ad ora per il foglio incorporato a inizio pagina. Unica differenza che non dovrà contenere al suo interno le dichiarazioni di style perchè già presenti nella pagina html che lo richiama.

Proviamo ad immaginare i vantaggi. Pensate di dover fare una modifica all'intero sito, magari semplicemente per cambiare un tipo di font che non vi piace più o sostituire un colore con un altro, sarà sufficiente apportare la modifica una sola volta a quel singolo foglio esterno per vedere poi immediatamente su tutte le pagine il risultato della modifica.

Comodo vero? Sono finiti i tempi in cui si passavano intere ore a rielaborare tutte le varie pagine per le modifiche e dove, immancabilmente, si finiva sempre col dimenticarne qualcuna.

Per creare questo foglio esterno sarà sufficiente ancora una volta un editor testuale, lo stesso che adoperate per html, meglio ancora se uno specifico per css. Vi ricordo che il wordpad, o notepad, a corredo di windows andranno benissimo, l'importante sarà salvare sempre in formato ascii, cioè testo puro privo di particolari formattazioni, avendo cura di fare in modo che il file così salvato sia di tipo css e non txt come normalmente avviene. Se qualcuno avesse difficoltà con questo tipo di operazione (rinomina del file), potrà leggere come fare su questa pagina.

Esaminiamo più attentamente il nostro foglio di Stile esterno:

```
p{
text-align: justify;
text-indent: 12px;
}
```

Si noterà che praticamente è lo stesso usato nello "style incorporato" con la sola differenza che non ha più gli elementi iniziali che ne dichiaravano lo stile: <style type="text/css">, questo perché trattandosi di un foglio esterno la dichiarazione viene già messa nel documento HTML che lo richiama facendo uso della seguente sintassi:

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

Avendo cura di posizionare questa riga di richiamo fra gli elementi (tags) <head> e </head> del file html. Per essere certi di non sbagliare mettetela come ultima riga di head. praticamente prima di </head><body>. Il nome_assegnato.css sarà il nome esatto che avrete deciso di usare per il vostro foglio di stile esterno. Evitate nomi lunghi e complessi ed evitate anche spazi vuoti e caratteri riservati quali: £ $ % & / (? nel nome che deciderete di usare.

Oltre al type è possibile specificare anche il media al quale si fa riferimento, dove *all* per tutti i dispositivi di visualizzazione, *screen* è usato per il browser sullo schermo, *print* per la stampante, *handheld* per dispositivi come i palmari. Volendo si possono specificare più di un solo media separandoli da virgole.

```
<link rel="stylesheet" href="nome_assegnato.css" type="text/css" media="screen">
```

Si possono avere più fogli di style esterni, è sufficiente richiamarli dalla pagina web allo stesso modo descritto sopra, uno di seguito all'altro:

```
<link rel="stylesheet" href="foglio_uno.css" type="text/css">
<link rel="stylesheet" href="foglio_due.css" type="text/css">
<link rel="stylesheet" href="foglio_tre.css" type="text/css">
```

Oppure dall'interno del foglio di style, sia interno che esterno, facendo uso di import ad inizio foglio, prima di definire qualsiasi altra regola e facendo precedere import dal simbolo chiocciola @.

```
@import "foglio_due.css";
@import "foglio_tre.css";
```

Un foglio di style esterno tipo potrebbe essere questo:

```
html, body{
        margin-left: .5cm;
        margin-right: .5cm;
        color: #000099;
        font-family: Verdana, Arial, sans-serif;
        font-size: 10pt;
        }

a:link{
        text-decoration: none;
        color: #009900 ;
        }

a:visited{
        text-decoration: none;
        color: Gray;
        }
```

```
a:hover{
        color: #ff0000;
        text-decoration: none;
        }

td{
        font-family: "MS Sans Serif";
        font-size: 12pt;
        }

p{
        text-align: justify;
        text-indent: 12px;
        }
```

Vediamo cosa è stato ridefinito, in modo da capire meglio e cominciare a familiarizzare con le varie sigle:

html, body tutto quello dichiarato all'interno di questo elemento { ... } sarà esteso a tutto il documento, per cui tutta la pagina web farà uso di:

font-family: *Verdana*, *Arial*, *sans-serif*;
font verdana, nel caso in cui questo non fosse installato nel PC del visitatore si cercherà di usare Arial, se anche questo dovesse mancare allora un font generico della famiglia sans-serif, dopo di che sarà assunto il font di default.
margin-left: *.5cm*; margin-right: *.5cm*;
Margine laterale dai bordi destro e sinistro: 0,5cm
color:*#000099*;
Colore del testo blu scuro
font-size: *10pt*;
Dimensione del font 10 punti
a:link{text-decoration: *none*; color: *#009900*;}
Link da visitare non sottolineati e di colore verde
a:visited{text-decoration: *none*; color: *Gray*;}
Link visitati non sottolineati e di colore grigio
a:hover{text-decoration: *none*; Color: *#ff0000*;}
Cambio colore al passaggio del mouse sopra il link, non sottolineati e di colore rosso
td{font-family: *"MS Sans Serif"*; font-size: *12pt*;}
Celle di tabelle con font Ms Sans Serif dimensione 12 punti
p{text-align: *justify*; text-indent: *12px*;}
Paragrafo con allineamento giustificato, indentizione 12 pixel
NOTA: un parametro composto da due o più parole come in questo esempio il nome del font, deve essere racchiuso da virgolette, font-family: *"MS Sans Serif"*; Quando si propone una scelta fra diversi tipi di fonts (uso di font-family), l'ultima scelta deve essere di una famiglia di fonts, in questo caso *sans-serif*.
In questa fase di apprendimento non si dovrebbe badare troppo al significato di tutte le varie sigle usate per l'esempio, meglio concentrarsi sul metodo di utilizzo, una volta capito il meccanismo le sigle da adoperare si vanno a leggere e si possono copiare dalla tabella dei parametri.

Le classi

Qualunque sia la nostra scelta, foglio di style esterno o incorporato, non è detto che si debba necessariamente ridefinire tutti o alcuni degli elementi (tags) di Html, anzi, lasciamoli pure stare così come sono e creiamo noi le varianti che ci servono facendo uso delle *classi*. Questo di far uso di classi è decisamente il sistema più comodo e pratico. Se ne possono creare quante e come ci pare e piace, così facendo si sfruttano tutte le potenzialità dei CSS (fogli di style).

E' possibile definire con estrema semplicità delle proprie classi di stile abbinandole allo stesso elemento di html, saranno poi queste classi ad essere richiamate dall'interno dei vari elementi (tags) di html. Pensate al fatto che lo stesso elemento <p> potrebbe richiamare quando una classe, quando un'altra classe a seconda delle esigenze. Come dire che si ottengono dallo stesso elemento risultati completamente diversi a seconda di quale classe viene ad esso associata.

Esempio: Definiamo una classe, per farlo è sufficiente editare un nome preceduto da un . punto, quale nome?

Non ha importanza, decidetelo voi, meglio se attinente con l'effetto che dovrà produrre in modo tale da essere ricordato con maggiore facilità quando si dovrà adoperarlo.

Nota: i nomi delle classi devono iniziare con un carattere alfabetico e non con un numero e non devono contenere spazi vuoti.

Vediamo praticamente come fare. Immaginiamo di volere qualcosa che serva per la nostra firma, ad esempio quella che io ho messo a fine pagina. Creiamo una classe alla quale assegneremo il nome firma che farà uso di un font piccolo ed inclinato, quindi stile italico, ed un colore rosso ben visibile.

Richiamiamo dal nostro editor il file .css se si trattava di un foglio di stile esterno, altrimenti portatevi all'interno della zona style, posizionandovi fuori da eventuali parentesi graffe. Possiamo ora descrivere la nostra nuova classe *firma* assegnandole i vari attributi:

```
.firma {
        font-family: Verdana, Arial, sans-serif;
        color: #ff0000;
        font-size: 9pt;
        text-align: center;
        font-style: italic;
        }
```

Da notare che fra il **.**(punto) iniziale e la scritta firma non ci sono spazi vuoti. Vediamo cosa fanno le singole voci inserite in questa nuova classe *firma*:

font-family: *Verdana, Arial, sans-serif*;
Significa usare il font *Verdana* (il primo dichiarato), se questo non fosse installato sul PC del visitatore si cercherà *Arial* (il secondo dichiarato), nel caso mancasse anche quello si passerà ad uno della famiglia *san-serif* e così via, si possono specificare fino a tre fonts alternativi, nel caso in cui nessuno dei tre fosse installato sarà visualizzato usando quello standard di default.

color: *#ff0000*;
Colore *rosso*, si possono esprimere i colori in formato esadecimale, come in questo caso, facendoli precedere dal segno pound # (cancelletto), oppure scrivendo il relativo nome del colore in lingua inglese. Anche *red* avrebbe prodotto lo stesso risultato.
Su questa pagina potete trovare i codici esadecimali per i colori più comuni.
font-size: *9pt*;
Dimensione del font: *9 punti*, potevamo esprimere il valore anche in pixel o in centimetri o anche in altri modi che vedremo in seguito. In pixel forse è meglio che in punti perchè si adatterà perfettamente alla risoluzione video, qualunque essa sia. Da notare che non ci sono limiti alle dimensioni a differenza dei font dichiarati in html, dove il valore max era 7. Se esprimete le dimensioni in punti (pt) non scendete mai sotto al valore 9 perchè i sistemi MAC non visualizzano font di dimensioni inferiori a 9pt.
text-align: *center*;
Allineamento del testo, in questo caso *centrato*, si poteva scegliere fra destro, sinistro o giustificato; adoperando i relativi termini in lingua inglese: left, right, justify

font-style: *italic*;
Stile Italico, potevamo scegliere anche normal o oblique.

Bene, la classe è pronta e non resta che richiamarla ogni volta che ci serve, vediamo come.

Molto semplice, ogni elemento (tag) di HTML supporta l'attributo class, quindi dovendo scrivere una parte di testo e volendo che questo faccia uso degli attributi appena inseriti nella classe firma, basterà assegnare la classe "firma" all'elemento (tag) prescelto, nel nostro caso un paragrafo:

```
<p class="firma"> ... </p>
```

Tutto ciò che sarà scritto all'interno di questo paragrafo rispetterà i parametri, anzi gli stili, della classe *firma*. Mi raccomando di non dimenticare la chiusura dell'elemento o le impostazioni di firma saranno estese a tutto il resto del documento. Così facendo gli altri elementi non subiranno alcuna variazione perchè ad essi non viene assegnata la classe firma come invece è stato fatto per questo elemento in particolare.

Avrete certamente capito che a questo punto qualsiasi elemento di html altro non sarà che un contenitore vuoto all'interno del quale poter inserire tutto quello che ci pare e piace. Potremo continuare ad usare altri con classi diverse ed avere così paragrafi formattati in modo completamente differente fra di loro. Basterà semplicemente cambiare il nome della classe facendogli richiamare una delle tante create per le varie occorrenze. Da questo modo di lavorare si coglie la vera versatilità e potenza dei fogli di style.

In pratica l'esempio:

```
<p class="firma">questo testo rispetta gli attributi della classe firma </p>
```

Produce questo il risultato:

questo testo rispetta gli attributi della classe firma

Ed infatti usa il font Verdana, è di colore rosso, ha dimensioni 9pt, è centrato ed ha lo stile italico.

Si possono anche combinare le varie classi nidificandole una dentro all'altra grazie ai vari contenitori, quali   per esempio, la cosa però non sempre è fattibile e per questo si raccomanda di acquisire prima una certa dimestichezza onde evitare pasticci indecifrabili.

A questo punto vi sarete resi conto che non è poi così difficile, possiamo creare tutte le classi che ci servono con i parametri e gli attributi più adeguati alle nostre esigenze e battezzarle con i nomi che vogliamo. Se all'interno del vostro sito, fatto di una o di cento pagine html, vi fosse anche una sola piccola parte di testo che non vi convince, sarà sufficiente creare una nuova classe ed assegnarla a quella parte di codice.

Ci sono anche classi che definiremo più rigide perchè associate a priori ad un particolare elemento o ad una lista ben definita di elementi html, in questo caso il nome della classe non produrrà alcun effetto se non quando viene richiamato dagli stessi elementi predefiniti. Vediamo un paio di esempi, volendo associare una particolare classe al solo elemento paragrafo. La sintassi per richiamare la classe resta la stessa: , ma nel foglio di stile è possibile stabilire fin da subito a quale tag la classe potrà essere associata.

```
p.destro {text-align: right;}
p.grassetto {font-weight: bold;}
```

In questo caso possono essere usate le classi *destro* e *grassetto* ma produrranno il loro effetto soltanto con l'elemento <p>

```
<p class="destro"> Questo paragrafo è allineato a destra.</p>
<div class="destro"> Questo div invece se ne infischia della classe
che allinea a destra.</div>
```

Volendo associare la stessa classe a più elementi html questi si possono dichiarare nel foglio di style separandoli da una virgola:

```
p.destro, div.destro {text-align: right;}
p.grassetto {font-weight: bold;}
```

In questo caso possono essere usate: la classe *destro* con l'elemento <p> e/o con quello mentre *grassetto* funzionerà soltanto con l'elemento <p>

```
<p class="destro"> Questo paragrafo è allineato a destra.</p>
<div class="destro"> Adesso anche questo div è allineato a destra.</div>
```

NOTA: è possibile richiamare da html più di una sola classe per lo stesso elemento, è sufficiente inserire all'interno di *class* i nomi delle classi separandoli da uno spazio vuoto, questo perchè così come tutti gli attributi di html anche class si adopera una sola volta all'interno dello stesso elemento.

```
<p class="destro grassetto">
```

E' possibile fare la stessa operazione anche dall'interno del foglio di style dichiarando per classi diverse gli stessi attributi, così facendo la classe già esistente si porterà dietro (eredita) tutte le proprietà della stessa classe dichiarata precedentemente.

```
p.destro {text-align: right;}
.grassetto, p.destro {font-weight: bold;}
```

In questo caso con la classe *destro* assegno al solo elemento <p> un allineamento a destra. Poi definisco una nuova classe di nome grassetto, da usare con qualsiasi elemento di html e separandola da una virgola richiamo anche la classe *destro* che avevo assegnato al tag <p>. In questo caso quando userò *destro* oltre ad allineare a destra il testo sarà anche in grassetto.

```
<p class="destro"> Questo p è allineato a destra + grassetto.</p>
<div class="grassetto"> Questo div ha solo il grassetto.</div>
```

Se vi sembra un po' complicato non preoccupatevi, facendo delle prove e vedendo i risultati a video si capirà meglio il meccanismo, alo momento imparate ad usare le classi semplici, il resto arriverà da solo.

I selettori ID (Identificatori)

Un selettore ID o identificatore, svolge funzione di etichetta di un contenitore, si possono cioè assegnare dei parametri e marcarli con un ID così che quando serviranno quei parametri, basterà richiamare il nome del selettore (ID) e con esso saranno richiamati tutti i valori a lui associati, valori specificati una sola volta nel foglio di style (interno o esterno che sia).

L'uso degli ID, così come quello delle classi, può essere associato a qualsiasi elemento valido di HTML anche se di norma lo si adopera con l'elemento vuoto Questo elemento (div), come detto in precedenza, di per sé non avrebbe molto significato, almeno non in HTML dove risulta essere un semplice contenitore vuoto che non provoca alcun effetto che non sia un ritorno a capo. Trova largo uso invece proprio con i CSS, specialmente per la gestione delle immagini, dei blocchi di testo e della struttura (layout) per impaginare.

Il suo uso rende il codice più pulito e comprensibile in quanto con il solo nome dell'identificatore è possibile richiamare tutti gli attributi ad esso associati, anche nel caso in cui questi siano molti.

Ancora una volta un esempio chiarirà meglio di qualsiasi discorso:

Definiamo all'interno di uno style un ID, per farlo è sufficiente assegnare un nome, nel nostro esempio: *weblink*, preceduto dal simbolo # pound (cancelletto), vediamolo:

```
<style type="text/css">
<!--

#weblink {

/* elenco parametri usati per il nostro esempio: */

font-family: Verdana, Arial, sans-serif;
color: #ff0000;
font-size: 10pt;
text-align: center;
font-style: italic;
}
-->
</style>
```

A prima vista si direbbe identico ad una classe, ed infatti è molto simile, io aggiungerei che forse è più limitato di una classe (sempre a prima vista), in quanto un selettore ID può essere adoperato una sola volta all'interno della stessa pagina, mentre una classe non ha praticamente alcun limite.

La cosa positiva è che si possono combinare classi e selettori ID, per cui evviva l'abbondanza che sarà sfruttata sicuramente al meglio dall'estro di ognuno di noi.

Ma ci sarà pure un motivo del perchè un *id* apparentemente simile ad una *classe* sia usabile una sola volta per pagina? Certo che c'è e lo vedremo più avanti parlando di box liquidi e layout.

Per tornare al nostro esempio, abbiamo inserito gli stessi attributi usati per la classe firma, per cui otterremo lo stesso identico risultato:

```
<div id="weblink">questo testo rispetta gli attributi del selettore weblink </div>
```

che non differisce molto da quella usata per le classi:

```
<div class="firma"> questo testo ecc. ecc </div>
```

Questo il risultato in entrambi i casi:

questo testo rispetta gli attributi del selettore weblink

Se vi ricordate infatti la classe firma produceva proprio questo stesso risultato.

Abbiamo detto che l'uso di *ID* è di solito lasciato per il posizionamento di oggetti grafici, non che non sia possibile posizionarli anche in altro modo e lo vedremo parlando di grafica. Per il momento preferisco non rischiare di confondervi le idee, per cui vediamo come fare con l'uso di ID, sarete poi voi a decidere quale potrebbe essere il sistema migliore a seconda delle vostre necessità.

Immaginiamo di voler posizionare un'immagine a 68 pixel dal margine superiore e 45 pixel dal margine laterale sinistro. Nel nostro esempio impostiamo i riferimenti a questi due parametri facendo uso di posizionamenti assolute e cioè:

```
<style type="text/css">
<!--

#weblink2 {

position: absolute;
     top: 68px;
    left: 45px;
  z-index: 10;
       }
-->
</style>
```

Adesso non resta che inserire il contenuto, immagini, testo o anche entrambi, all'interno dell'elemento associando al selettore ID "weblink2". In questo modo il contenuto di DIV risulterà posizionato con estrema precisione a 68 pixel dal bordo superiore ed a 45 pixel da quello laterale sinistro con una priorità di visualizzazione uguale a 10.

Ancora una volta capirete meglio il significato di queste sigle più avanti, quando saranno spiegate in modo più accurato, per il momento è sufficiente capire il meccanismo, per cui la sintassi corretta in html è questa:

```
<div id="weblink2"> ciao ciao </div>
```

Se avete capito come funziona avrete anche capito che ho dovuto chiamarlo weblink2 perchè essendo possibile usare lo stesso ID una sola volta per pagina con id weblink è già stato usato sopra, e poi che il mio ciao ciao si trova a 68 pixel dal bordo superiore e a 45 pixel dal bordo laterale sinistro, per cui scorrete in alto ad inizio pagina, fate bene attenzione a quella zona laterale sinistra e potrete notare la piccola scritta ciao ciao, notate anche che questa si trova sopra all'immagine Web-Link, un posizionamento impossibile con il solo HTML ma reso estremamente semplice dall'uso dei CSS.

Immagini nel Testo

Quando si devono inserire delle immagini insieme con il testo, queste risultano essere in linea col testo stesso. I Fogli di Style permettono di cambiare questo allineamento tramite la proprietà vertical-align che accetta i seguenti parametri: *baseline*, *top*, *middle*, *bottom*, *sub*, *super*, *text-top* e *text-bottom*. Personalmente trovo che il risultato non sia soddisfacente perchè l'iimagine è di solito più alta di una riga di testo e il risultato con le righe che precedono e quelle che seguono non è dei più gradevoli; ma è soltanto un mio parere e come tale va considerato.

Con la proprietà display è possibile cambiare il flusso naturale passando da linea a blocco e viceversa, quindi display: block; trasforma l'immagine in un blocco e genera un ritorno a capo permettendo al testo di disporsi sopra e sotto ma non di lato. display: inline; fa l'esatto contrario ripristinando il normale flusso.

```
img {
display: block;
}
```

Per avere il testo che circonda l'immagine si fa uso della proprietà float: con i parametri *left* o *right* questi permettono al testo di disporsi intorno all'immagine a sinistra o a destra.

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

Questo il risultato:

questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left n serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di

float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: left

Con l'attributo margin è possibile distanziare l'immagine dal testo agendo su ognuno dei quattro lati perimetrali.

```
img {
float: left;
margin: 15px 15px 15px 15px;
}
```

Questo il risultato:

questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left n serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di

float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left

questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float : left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: left questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: left

```
img {
float: right;
margin: 15px 15px 15px 15px;
}
```

Questo il risultato:

questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di

float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right

Mettendo in pratica l'uso degli identificatori ID visto in precedenza, si procede ad assegnare un ID al nostro elemento img che poi sarà richiamato da html:

```
img#testosx {
float: right;
margin: 15px 15px 15px 15px;
}
```

```
<img id="testosx" src="web-link.gif" width="245" height="133" alt="">
```

Il testo con float right non è bello a vedersi, lo si può aggiustare assegnando al testo il parametro per la giustificazione, <p align="justify"> risulterà così molto ben ordinato:

questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di

float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini facendo uso di float: right questo testo non serve a nulla se non a far vedere l'effetto con testo e immagini

Da notare la disposizione del testo intorno all'immagine ovale, si dispone formando un rettangolo, l'immagine infatti, anche se apparentemente ovale, ha una forma non visibile che può essere soltanto rettangolare o quadrata. Non è infatti possibile creare immagini rotonde, si tratta di un effetto ottenuto impostando delle trasparenze negli angoli, se inseriamo il bordo si vedrà meglio il suo reale perimetro, lo stesso che il testo circonda.

L'effetto float potrebbe proseguire oltre il blocco definito creando effetti non desiderati, per mettere fine a questa impostazione è conveniente usare la proprietà clear nel blocco o immagine successivo. clear accetta come parametri: *left*, *right* e *both* dove con both si annullano entrambi. clear: both

Grafica: posizionare gli oggetti Se avete letto la parte dedicata ai selettori ID vi sarete resi conto che i CSS non servono solo per assegnare stili al testo ma sono quanto di meglio si possa avere per il posizionamento degli oggetti sulla pagina web.

Per oggetti si intende tutto quello che fa parte di una pagina: immagini, testo, tabelle, forms e qualsiasi altra cosa venga racchiusa in un contenitore virtuale che prende il nome di box e che viene assegnato all'elemento (tag) di html .

I metodi per gestire questi contenitori virtuali ancora una volta sono diversi, ognuno di voi adopererà quello che preferisce. A prima vista sembrerebbe più semplice fare uso di un selettore ID che contenga la posizione del contenitore ed associare poi il nome del selettore al contenitore stesso, ma è anche vero che un risultato identico lo si avrebbe specificando la sua posizione nello style in linea, assegnando cioè all'elemento interessato i vari parametri di posizionamento in modo diretto.

Vi ricordate le classi?

Si creava una classe perché poteva poi essere richiamata più volte all'interno della stessa pagina. La classe va benissimo per lo style applicabile a testi, tabelle, links che su una pagina possono ripetersi in più punti. Dovendo invece posizionare qualcosa questo posizionamento va adoperato una sola volta, provate ad immaginare se la classe con i posizionamenti venisse usata da più di un solo elemento, tutti quelli che ne fanno uso si sovrapporrebbero convogliando nello stesso punto, uno sull'altro dando così origine a qualcosa di molto simile ad un pasticcio visivo.

E' forse questa la spiegazione per il fatto che un selettore ID usato per posizionare può essere adoperato una sola volta all'interno della stessa pagina web?. E' probabile, ma è soltanto una mia supposizione.

Penso comunque che non possa essere visto come una limitazione, al contrario. Credo sia una garanzia per impedire di commettere errori di questo genere. Si tratta di maggiore versatilità dal momento che si possono richiamare sia classi che ID dallo stesso contenitore html.

Adesso provate a creare un selettore ID con questi parametri:

```
<style type="text/css">
<!--
#weblink {
        position: absolute;
        top: 1300px;
        left: 50px;
        z-index: 1;
    }
 -->
</style>
```

Come per le classi, se volessi avere la certezza di usare il selettore #weblink soltanto con un contenitore di tipo * * dovrei dichiararlo con div#weblink, così facendo qualsiasi altro tag di html diverso da *div* non avrebbe procurato alcun effetto. Vediamo come richiamarlo da html:

```
<div id="weblink"> codice html </div>
```

Oppure è possibile assegnarlo direttamente in line adoperando gli stessi parametri:

```
<div style="position: absolute; left: 50px; z-index: 1; top: 1300px">...</div>
```

In entrambi i casi il risultato sarà perfettamente identico, entrambi visualizzeranno quello che si trova all'interno del contenitore   rispettando gli attributi di stile impostati. Nel nostro caso usiamo una immagine grafica, questa sotto:

che sarà posizionata a 1300 pixel dal bordo superiore e 50 pixel dal bordo laterale sinistro, con uno z-index, cioè un livello di profondità, uguale a 1.

Guardate qui sotto cosa è possibile fare giocando sul solo parametro z-index che riguarda la profondità degli oggetti così dichiarati. Ho assegnato al *div* che contiene il testo una profondità *10*, al *div* dell'immagine con sfondo giallo una profondità *5* e al div dell'immagine con sfondo verde una profondità *15*. Notate come il testo si trovi sotto o sopra le due immagini a causa del suo valore 10 che è maggiore di 5 ma minore di 15, da questi valori dipende l'ordine di profondità. Il solo browser IE gestisce anche uno z-index negativo che in questo caso ci

avrebbe fatto risparmiare un div ma che non garantiva compatibilità e di conseguenza l'essere visto da tutti i browser.

In HTML non sarebbe stato possibile ottenere nulla di tutto questo, col nostro testo e le due immagini avremo potuto soltanto girargli intorno.

A questo punto ogni altro esempio sarebbe superfluo, è fin troppo evidente quali siano le potenzialità ed i vantaggi derivati dall'utilizzo di questa tecnica.

Immagini trasparenti potrebbero essere sovrapposte totalmente o parzialmente dando vita ed effetti speciali. Con l'uso di un linguaggio di script quale JavaScript, si potrebbero nascondere o scoprire questi contenitori dando vita a delle pagine dinamiche, cioè vive, che cambiano a seconda di quello che fa il nostro visitatore, da questo connubio nasce la sigla DHTML (Dynamic HTML). E' grazie alla combinazione di questi due sistemi (CSS e JavaScript) che sono venuti meno quelli che erano considerati i limiti del linguaggio html. Adesso ci sono nuove costruzioni molto più "solide" anche se a sorreggerle resta proprio il buon "vecchio" html che a quanto pare regge, ed io aggiungo che regge molto saldamente!

Come spesso accade non è tutto oro quello che luccica, così anche per questa tecnica che apparentemente sembrerebbe offrire soltanto vantaggi, ci sono invece alcuni problemi. Mi riferisco al posizionamento degli elementi sullo schermo usando coordinate percise.

Chi ne fa uso prende come modello il proprio monitor e la risoluzione grafica su di esso impostata. Ma se il visitatore non ha le stesse identiche impostazioni ecco che quello che vedrebbe non sarebbe proprio la stessa cosa che pensavate di fargli vedere.

Provate ad immaginare di posizionare un qualcosa ad una distanza di 400 px dal bordo superiore, questa misura essendo fissa sarà sempre rispettata da qualsiasi risoluzione video. Il codice rimanente si disporrà nella pagina seguendo le dimensioni della finestra del browser, al variare di questa si ridispone automaticamente mentre quello che avete posizionato con riferimenti assoluti non si sposterebbe di un solo pixel dando così vita ad impaginazioni certamente differenti e non volute.

Fate dunque molta attenzione quando adoperate i posizionamenti assoluti. Per capirli meglio gli ho dedicato una pagina: posizionare il box e vi dico che la soluzione esiste e si chiama Box liquido che permette di adattare il tutto dinamicamente in base alle varie risoluzioni grafiche.

il Box Model.

Se siete dei principianti in assoluto vi consiglio di saltare questa parte della guida passando direttamente ai links. Potrete sempre tornare dopo aver capito i meccanismi ed aver acquisito una certa familiarità con l'uso dei css.

Una delle cose più importanti introdotte con i CSS è il box model che permette di creare una struttura (layout) fatta da rettangoli e/o quadrati, praticamente colonne e righe con o senza bordo che servono per poter impaginare i dati quali: testi, immagini o qualsiasi altra cosa che potrebbe far parte di una pagina web.

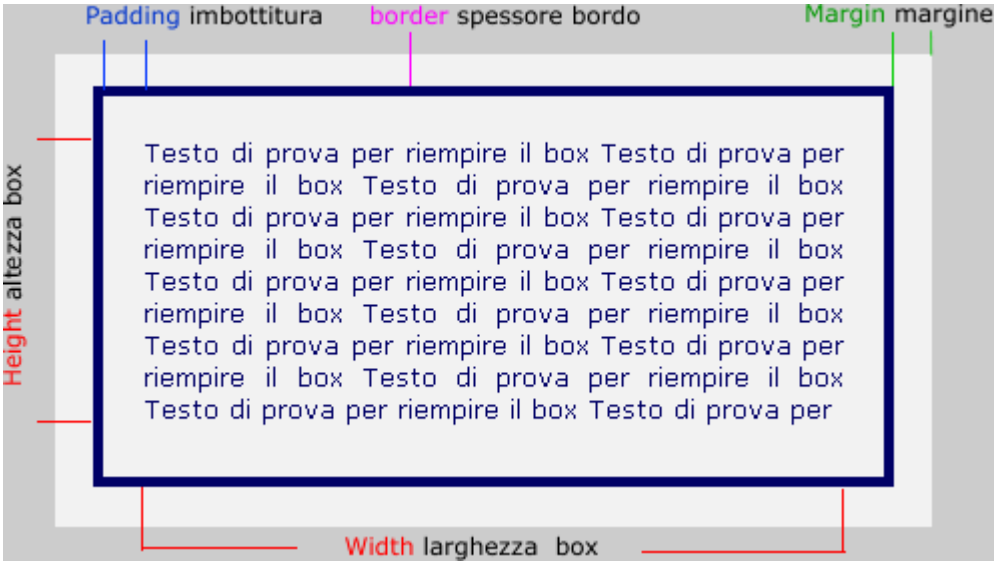
Questo il codice in css per definirlo:

```
#box {
width: 350px;
height: 150px;
padding: 20px;
border: solid 5px;
margin: 20px;
```

```
}
```

Dove: width e height sono rispettivamente larghezza ed altezza del contenitore e di conseguenza va proporzionato rispetto a quello che potrà essere inserito al suo interno.

padding lo spazio fra i contenuti ed il bordo del box.
border lo spessore del bordo perimetrale del box.
margin la distanza del box dagli altri elementi della pagina.



Questo il codice html per richiamarlo all'interno della pagina:

```
<div id="box">Testo di prova per riempire il box... </div>
```

Questo il risultato a video:

Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per

Da notare che la larghezza e l'altezza totale di un box sono dati dalla somma dei contenuti più quello del padding più quello dello spessore del bordo. Nel nostro caso lo spazio occupato realmente sarà: 350 + 20 + 20 + 5 + 5 = 400 pixel per la larghezza e 150 + 20 + 20 + 5 + 5 = 200 pixel in altezza. Praticamente le misure specificate (350 x 150) si riferiscono ai contenuti.

Questo è vero per molti browser tranne alcuni, per esempio IE (internet explorer), il famoso browser della Microsoft, il più adoperato ma anche quello che meno di altri rispetta le direttive del W3C. Li chiameremo bugs e di questi ce ne sono diversi che si vanno ad aggiungere a molte proprietà dei CSS che non sono supportate. Pazienza, noi naviganti confidiamo nelle versioni future e nel frattempo chiediamoci come mai noi creatori di siti web ci diamo tanto da fare per adeguarci ai nuovi metodi quando poi certe case produttrici di browser possono fare come gli pare. Ma questo è un discorso che qui non c'entra molto.

Se si tratta di un solo box probabilmente nessuno si accorgerebbe di eventuali differenze ma dovendo creare un layout formato da più box ecco che le differenze si noterebbero. Qualche pixel in più o qualche pixel in meno sarebbe sufficiente per scombinare tutta l'impaginazione, per fortuna si può rimediare, vediamo come.

Internet Explorer nelle versioni 5 e 5.5 ma anche per la versione 6 (se opera in *quirks mode*) considera la larghezza e l'altezza dei contenuti sottraendo dalle misure specificate quelle dello spessore del bordo e dello spazio padding. Nel nostro caso il box esterno sarà realmente 350 x 150 Pixel ma il box dei contenuti si rimpicciolisce diventando: 350 - 20 - 20 - 5 - 5 = 300 pixel per la larghezza e 150 - 20 - 20 - 5 - 5 = 100 pixel per l'altezza.

In questo esempio la differenza sarebbe notevole se visto con un browser che non ha bugs ed un browser IE V.5, Considerando che IE sottrae bordo e padding dalla larghezza e dall'altezza totale, si potrebbe risolvere creando due box, uno interno all'altro dove nel box esterno non si specificano ne bordi ne padding che specificheremo nel box interno.

```
#box-esterno {
width: 400px;
height: 200px;
}

#box-interno{
padding: 20px;
border: solid 5px;
margin: 20px;
}
```

Questo il codice html per richiamarlo all'interno della pagina:

```
<div id="box-esterno"><div id="box-interno">  Testo di prova per riempire il box...
</div></div>
```

Questo il risultato a video:

Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per riempire il box Testo di prova per

Il risultato a video è lo stesso per tutti i browser ed il bug di IE non assegnerà al box dei contenuti misure differenti perchè se è vero che il problema nasce quando nel box vengono definite misure di larghezza altezza insieme ai bordi o al padding, il box interno non ha misure dalle quali sottrarre bordi e padding mentre nel box esterno non ci sono bordi e padding da sottrarre alle misure.

PS: se vedete i due box diversi frà di loro il vostro browser è uno di quelli affetti da bug del box model. Ne approfitto per ricordare che tenere aggiornato il proprio browser alla sua ultima versione offre garanzie non solo per bug di visualizzazione ma anche di sicurezza, per cui andrebbe sempre aggiornato visto poi che l'operazione è completamente gratuita.

Posizionare il Box Model.

Abbiamo visto che un box è praticamente un contenitore che inserito in un punto qualsiasi della pagina ne segue il suo flusso naturale generando un ritorno a capo come tutti gli elementi di blocco, ne consegue che più box sarebbero disposti verticalmente uno di seguito all'altro. A causa di questa caratteristica diventa impossibile creare dei layout elaborati usando più box senza fare uso di una proprietà dei fogli di style dal nome position che permette di posizionare il box in modo diverso da quello *naturale*.

Gli attributi ammessi per la proprietà position sono: *static*, *relative*, *absolute* e *fixed*.

Vediamoli per scoprire e capire a cosa servono.

static è il posizionamento predefinito, quello naturale che segue il flusso nella disposizione della pagina. Il suo posizionamento è nel punto preciso in cui viene creato o richiamato. Definiamo 3 box, tutti uguali, cambia soltanto il colore di fondo:

```
#box1 {
width: 400px;
height: 20px;
border: solid 1px;
color: black;
background-color: #ffff99;
}
```

Inseriti nella pagina seguendo il normale flusso di disposizione:

```
<div id="box1">blocco 1 inserito nel normale flusso </div>
<div id="box2">blocco 2 inserito subito dopo l'istruzione per il blocco 1</div>
<div id="box3">blocco 3 inserito subito dopo l'istruzione per il blocco 2</div>
```

Producono questo risultato:

blocco 1 inserito nel normale flusso
blocco 2 inserito subito dopo l'istruzione per il blocco 1
blocco 3 inserito subito dopo l'istruzione per il blocco 2

Notate come i blocchi inseriti nella pagina seguono il flusso naturale disponendosi verticalmente pur non avendo inserito alcun ritorno a capo, questo perchè il tag , così come tutti gli elementi di blocco, provoca automaticamente un ritorno a capo.

Non avendo impostato alcun margin i bordi risultano incollati fra loro. L'esempio sotto è lo stesso box con margin impostato a 5 pixel. Ricordate quanto detto precedente riguardo al box model? si fa riferimento al box interno, per cui i 5 pixel che distanziano i vari box sono 3 pixel più i 2 pixel esterni che fanno da bordo.

blocco 1 con margin inserito nel normale flusso
blocco 2 con margin inserito subito dopo l'istruzione per il blocco 1
blocco 3 con margin inserito subito dopo l'istruzione per il blocco 2

questo il codice per definirlo nel foglio di style CSS.

```
#box1 {
width: 400px;
height: 20px;
border: solid 1px;
margin: 5px;
color: black;
background-color: #ffff99;
}
```

Il codice si riferisce al box1 ma la definizione per gli altri due box è praticamente la stessa, varia soltanto per il nome: #box2 e #box3 ed il relativo colore di sfondo: background-color.

L'attributo margin accetta anche come valore auto che posizionerebbe il blocco al centro del flusso visto che assegnerebbe automaticamente lo stesso margine ad entrambi i lati.

blocco 1 con margin auto inserito nel normale flusso

Da notare che ad essere centrato è il blocco e non il suo contenuto, da notare anche che per avere un bordo nel foglio di style non basta specificare il solo spessore (*1px*), si deve specificare anche un possibile attributo che identifica il tipo di bordo usato, in questo caso *solid*, diversamente il bordo non risulterà visibile.

Quando il contenuto del box supera le sue dimensioni, per esempio inserendo un'immagine più larga del box definito, si hanno comportamenti differenti a seconda del tipo di browser, tanto per cambiare, dove ognuno di questi interpreta a modo suo. Alcuni allargano il box per adattarlo alla dimensione del suo contenuto, altri fanno uscire l'immagine dal box.

E' possibile regolare questo comportamento al fine di unificarlo su tutti i browser grazie alla proprietà overflow che accetta i seguenti possibili attributi: *visible*: predefinito, *hidden*: nascosto, *auto*: appaiono le barre di scorrimento soltanto se necessario, *scroll*: appaiono le barre di scorrimento sempre, anche se non necessitano.

overflow visible: predefinito, interpretato in modo differente dai vari browser.

overflow hidden: nascondi le barre e taglia l'immagine alle dimensioni del blocco.

overflow auto: taglia l'immagine ed inserisci soltanto le barre di scorrimento necessarie.

overflow scroll: taglia l'immagine ed inserisci entrambe le barre di scorrimento anche se non necessarie.

relative è il posizionamento modificato rispetto alla posizione predefinita, praticamente un *offset* che consente di spostare il box rispetto alla sua naturale origine, senza che questo spostamento incida o alteri il posizionamento degli altri elementi (oggetti) facenti parte della stessa pagina.

blocco 1 con margin inserito nel normale flusso

blocco 2 position relative *12px* top e *20px* left

blocco 3 con margin inserito subito dopo l'istruzione per il blocco 2

Notate come il *blocco 2* sia spostato di 12 pixel a partire dal margine alto e di 20 pixel a partire dal margine sinistro rispetto alla sua posizione naturale, senza che gli altri due box subiscano alcuna variazione di posizione. Questo perché position relative agisce solo sul box che ne fa uso senza alterare il normale flusso di tutto il resto.

E' possibile specificare la posizione relative usando gli attributi *top*, *left*, *bottom* e *right*. Rispettivamente per posizionarlo a partire dall'*alto*, da *sinistra*, dal *basso* e da *destra*. Sono ammessi anche numeri negativi che posizionerebbero il blocco in direzione contraria a quella di numeri positivi.

Questo il codice nel css per definirlo.

```
#box2 {
position: relative;
top: 12px;
left: 20px;
width: 450px;
height: 20px;
border: solid 1px;
margin: 5px;
color: black;
background-color: #ffccff;
}
```

Nell'esempio sopra, il blocco 2 risulta sovrapposto al blocco 3 oscurandolo in parte, è possibile stabilire l'ordine di sovrapposizione con la proprietà z-index che accetta solo valori numerici positivi, il numero più alto è quello in primo piano e copre il numero più basso.

Adesso facciamo in modo che sia il blocco 3 a risultare sovrapposto al blocco 2 perchè gli viene assegnato un valore z-index di *1* che è superiore al valore dello z-index *0* (di default è impostato a 0) del blocco 2.

questo il codice nel css per definirlo.

```
#box3 {
position: relative;
z-index: 1;
width: 450px;
height: 20px;
border: solid 1px;
margin: 5px;
color: black;
background-color: #ccffff;
}
```

Questo il risultato

blocco 1 con margin inserito nel normale flusso

blocco 2 position relative 12 px top e 20 px left

blocco 3 con z-index 1 superiore al blocco 2

absolute è il posizionamento assoluto, cioè è possibile posizionare il blocco in un qualsiasi punto della pagina, a differenza del posizionamento relativo che abbiamo visto non alterava gli altri oggetti, pur condizionandoli con la sua presenza, absolute farà in modo che il blocco così

definito risulti come se non facesse parte della pagina, praticamente non inciderà in alcun modo con gli altri oggetti, una specie di sovrapposizione alla pagina stessa.

E' possibile specificare la posizione absolute usando gli stessi attributi visti per relative *top*, *left*, *bottom* e *right*. Rispettivamente per posizionarlo a partire dall'*alto*, da *sinistra*, dal *basso* e da *destra*. Questa volta però il punto di riferimento è al primo blocco progenitore non statico e se non esiste, all'elemento <html>. Praticamente i margini della finestra del browser.

Se non viene specificato alcun valore di posizionamento il blocco assumerà valore assoluto riferendosi al punto in cui viene inserito dal normale flusso del codice. Su questa pagina è possibile vedere un esempio che chiarirà tutto quanto in modo più semplice.

fixed non supportato dai browser IE è simile al posizionamento assoluto ma il riferimento è sempre e solo alla finestra del browser, quando la pagina scorre i vari box così definiti restano fissi nella loro posizione specificata.

il Box Liquido.

Abbiamo visto come si creano i box e come si posizionano correttamente, esiste però una tecnica che prende il nome di box liquido che consentirà al box di adattarsi alla pagina ridimensionandosi automaticamente, proprio come fosse un liquido versato in un recipiente che ne assume la forma, da qui il nome box *liquido*.

Dovendo posizionare un box al centro della pagina abbiamo visto che è sufficiente impostare il valore *auto* alla proprietà margin

```
#box-centrato{
margin: auto;
}
```

```
<div id="box-centrato">Testo di prova per riempire il box...</div>
```

Su questo link il risultato a video

E' possibile eliminare i bordi perimetrali impostando nel body un margin ed un padding di *0 pixel*

```
body {
padding: 0;
margin: 0;
      }

#box-centrato{
margin: auto;
      }
```

Su questo link il risultato a video.

Oppure decidere per un bordo fisso (o in percentuale) agendo sul padding nel body per un box liquido e flottante.

```
body {
padding: 5% 5%;
margin: 0;
}
```

```
#box-centrato{
        margin: 0;
        width: 100%
        }
```

Su questo link il risultato a video.

Purtroppo però certi valori sono male interpretati (o per nulla supportati) da parte di alcuni browser.
Per evitare, o meglio per arginare, questa errata interpretazione è possibile adoperare alcuni "*trucchetti*", che rendono la visualizzazione quanto più possibile uguale per tutti i vari browser.

Per esempio, il browser IE versioni precedenti centra il box assegnando il valore center alla proprietà text-align riservata al testo, è possibile allora impostare questa proprietà nel body, per poi ripristinare il normale allineamento del testo nel box con text-align: left, come nell'esempio sotto.

```
body {
padding: 0;
margin: 0;
text-align: center; /* per il browser IE 5.5 */
}

#box-centrato{
margin: auto;
text-align: left; /* ripristinato allineamento del testo */
}
```

Si conclude il box liquido con la centratura verticale, anche se non sempre è consigliabile per via di un tipo di browser che taglia la parte superiore del box nel caso in cui questa avesse dimensioni maggiori della finestra del browser che lo visualizza.

```
#box-centrato{
position: absolute;
height: 200px;
top: 50%;
margin-top: -100px;

}
```

Si noti la tecnica utilizzata nel css: il box ha una dimensione in altezza (height) di 200 pixel, viene posizionato il suo margine superiore al centro della finestra del browser (top 50%) e poi viene impostato il suo margine superiore con un numero negativo che ne provoca lo spostamento verso l'alto che per l'esatta centratura deve essere della metà esatta dei pixel usati per la dimensione del box in altezza(200), in questo caso -100px, il box risulterà perfettamente centrato in senso verticale come è possibile vedere su questo link esempio.

I Layout fissi e liquidi.

Un layout altro non è che una struttura per contenere i dati. Si tratta in pratica di usare box fissi e/o box flottanti per impaginare tutto quello che può far parte di una pagina web.

Facendo uso delle informazioni e degli esempi visti in precedenza per il box liquido è possibile creare una struttura molto versatile, in questo esempio due colonne: una per il menù a sinistra e l'altra per la pagina principale. Una testata con titolo ed un margine inferiore. Lo possiamo vedere su questo link esempio.

Questo il codice nel foglio di style CSS, già tenuto conto del bug per i browser IE:

```
body {
        text-align: center; /* IE5.x */
        padding: 0;
        margin: 0;
        }

#telaio {
        width:100%;
        margin: 0px;
        border:1px solid gray;
        line-height:150%;
        color: black;
        background-color: #ffffcc;
        text-align: left; /* IE5.x */
        }

#testata {
        font-size: 130%;
        }


#fondo {
        font-size: 85%;
        text-align: center;
        }


#testata, #fondo {
        padding:0.5em;
        color:white;
        background-color: #cc9900;
        clear:left;
        }


#sinistro {
        float:left;
        width:160px;
        margin: 0px;
        padding: 5px;
        }

#centrale {
        margin-left: 170px;
        border-left:1px solid gray;
        padding:10px;
        }
```

Questo il codice HTML:

```
<body>


  <div id="telaio">


    <div id="testata">... Testata ...</div>
```

```
    <div id="sinistro">... Sinistro...</div>

    <div id="centrale">... Centrale ...</div>

    <div id="fondo">... Fondo ...</div>

  </div>

</body>
```

Su questo link il risultato a video

In questo esempio riassuntivo sono state introdotte molte delle cose trattate fino a questo momento per quanto riguarda il box model. Anche i parametri sono stati espressi volutamente con unità di misura differenti al solo scopo di rendere meglio l'idea di cosa sia possibile fare con l'uso dei fogli di style. Ognuno di voi potrà modificarlo a proprio gusto ed esigenza.

I links

Abbiamo visto come i CSS siano in grado di cambiare attributi al testo, come e con quale facilità sia possibile posizionare con estrema precisione oggetti grafici e testuali. Esiste anche un'altra novità introdotta dai CSS e riguarda la gestione dei links, si tratta di pseudo classi dinamiche.

La tanto odiata sottolineatura e la possibilità di cambiare colore o tipo di font sono soltanto alcune cose che possono essere gestite dai CSS. Compresi i cambiamenti dinamici nel momento esatto in cui il cursore del mouse passa sopra al link.

Prima di tutto vediamo come si adoperano perché differiscono leggermente da quanto appreso fino a questo momento. Anche i Link possono essere dichiarati in una pagina, in un foglio di stile esterno o direttamente in linea. Questa la sintassi:

```
<style type="text/css">
<!--

a:link    {text-decoration: none; color: brown;}
a:visited {text-decoration: none; color: green;}
a:hover   {color: red;}

-->
</style>
```

Dove a:link sono i link ancora da visitare (da cliccare). a:visited sono invece i links visitati (già cliccati) e a:hover si riferisce all'evento del mouse nel momento preciso in cui viene posizionato sopra il link.

Risultato: sono un link privo di sottolineatura e di colore marrone Provate a passarci sopra col cursore del mouse, noterete un cambio di colore, in questo caso rosso. Togliete il mouse ed il link tornerà del colore marrone impostato per il link. Adesso fate click su quel link, essendo visitato di conseguenza il suo colore passa al verde. Tutto come da impostazioni.

Non vi piace il colore rosso?

Sostituite nella pseudoclasse a:hover il parametro *red* (rosso in inglese) con qualsiasi altro nome "valido" di un colore. Allo stesso modo divertitevi a cambiare il colore dei link visitati e/o di quelli da visitare. I colori possono essere espressi anche con il relativo codice esadecimale preceduto dal simbolo # (pound o cancelletto) vedi anche la sezione dedicata ai parametri di questa stessa guida.

Il rosso viene rappresentato con #ff0000, dove le sei cifre vanno viste come tre coppie ben distinte di valori esadecimali ed esprimono i colori nella forma RGB, (rosso - verde - blu) dove i valori ammessi sono rappresentati da numeri che vanno da 0 a f (16) in notazione esadecimale.
Per cui ff = 16 x 16 = 256 (max valore ottenibile), abbiamo tre coppie ff ff ff = 256 x 256 x 256 = 16,7 Milioni di colori. Non c'entra molto con i links dei css ma almeno sapete il perchè di quelle sigle usate per impostare i colori.

Avete capito bene: *16,7 Milioni* di colori, tutti quelli che la vostra scheda video è in grado di visualizzare, sfumature comprese.

Anche per i links, una volta capito il meccanismo, si potranno combinare i vari attributi fino a creare degli effetti di notevole impatto, il tutto con estrema semplicità e senza ricorrere a programmazione avanzata. Da fare molta attenzione a non invertire l'ordine di dichiarazione:

1. link
2. visited
3. hover

Si deve rispettare questo ordine per evitare conflitti, se per esempio si invertisse *hover* con *visited* passando sopra ad un link già visitato non si avrebbe l'effetto hover del cambio di colore ma il colore fisso assegnato a visited che essendo posizionato dopo sovrascriverebbe l'effetto precedente.

Ci sarebbero altre due pseudoclassi dinamiche: focus ed active che possono essere applicate ad altri elementi di html oltre a link, non essendo però supportate dal browser IE è forse meglio non farne uso.

Ecco un altro esempio:

```
<style type="text/css">
<!--

a:link    {text-decoration: none; font-size: 10pt;}
a:visited {text-decoration: none; font-size: 10pt;}
a:hover   {color: red; font-size: 12pt;}

-->
</style>
```

Risultato: sono un link non sottolineato Se provate a passarci sopra col puntatore del mouse noterete che il font cambia dimensione, passando dai 10pt (punti) del link da visitare ai 12pt del link hover e di colore rosso. Così facendo avrete dato vita ad una vera e propria animazione. Notate che tutto il testo viene spostato per fare posto al font di dimensioni maggiori. Cliccate su questo esempio e leggete i consigli, tanto per avere le idee ancora più chiare.
E' anche possibile avere links all'interno della stessa pagina web con colori e comportamenti diversi. Sarà sufficiente fare uso di classi diverse da assegnare ai vari links:

```
Esempio 1
esempio 2
esempio 3
```

Le classi per i link si creano in maniera leggermente diversa ma molto simile alle classi viste precedentemente:

```
<style type="text/css">
<!--

a.xxx:link      { color: #cc66ff; text-decoration: none; }
a.xxx:visited   { color: #cc66ff; text-decoration: none; }
a.xxx:hover     { color: #ffcc00; text-decoration: none; font-size: 14pt;}

-->
</style>
```

Notate la sintassi, il . punto e i : due punti, dove *xxx* è il nome assegnato alla classe. Il suo richiamo da html segue la stessa sintassi degli altri elementi di html:

```
<a class="xxx" href="vostro link">esempio 3</a>
```

Risultato:

```
esempio 3
```

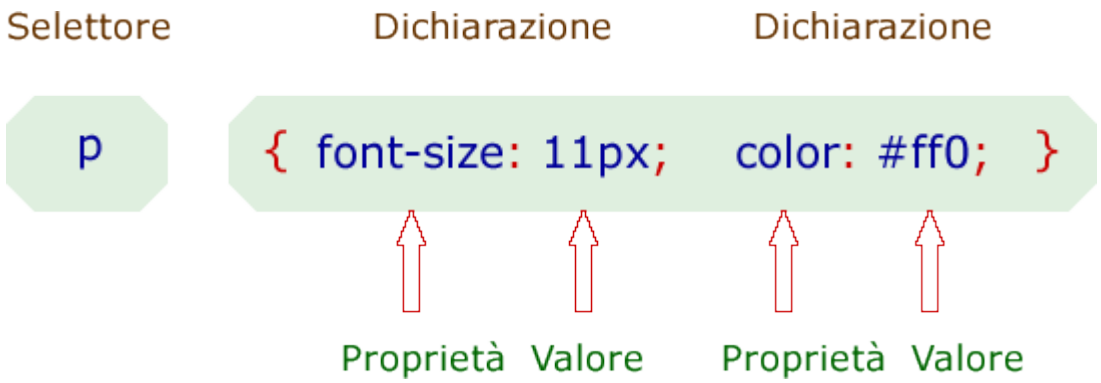
Gli attributi per gli stili Sono davvero molti i possibili attributi e non credo che riuscirei a descriverli tutti. Alcuni sono di raro utilizzo, altri supportati dai vari browser in modo differente. In queste pagine parleremo soltanto di quelli che si adoperano con una certa frequenza e che funzionano con tutti i browser di ultima generazione.

Lascio due links di sicuro interesse a chi di voi volesse approfondire questo argomento, si tratta delle specifiche del W3C, dove troverete tutti gli attributi supportati dai CSS1 e delle specifiche per i CSS2 . Se avete l'abitudine di navigare con un browser datato fate subito l'aggiornamento alla sua ultima versione, non costa nulla e vi darà maggiori garanzie dal punto di vista della sicurezza, oltre che supportare sicuramente meglio i vari CSS.

Prima di passare in rassegna le categorie di attributi vorrei ricordare ancora una volta la loro sintassi a prescindere da quale dei tre modi, in linea, nella pagina, nel foglio esterno, deciderete di usare.

Abbiamo detto che gli attributi devono essere inseriti all'interno di parentesi graffe { } e laddove in HTML verrebbe usato un "=" (uguale) nei css viene invece usato ":" (due punti).

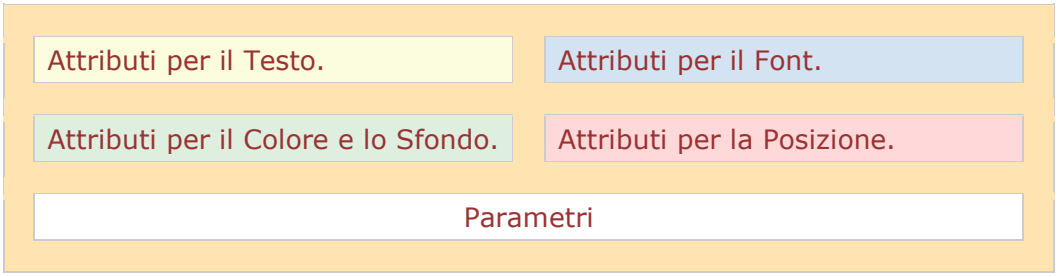
Argomenti consecutivi sono separati da un ";" (punto e virgola) invece che dalla "," (virgola). Inoltre molti argomenti sono divisi da "-" (trattino centrale) che fa parte integrante del nome stesso.



Gli spazi vuoti ed i ritorno a capo non hanno alcuna influenza.

```
selettore {
  proprietà: valore;
  proprietà: valore;
}
```

Per i parametri e gli attributi fare riferimento a questa tabellina.

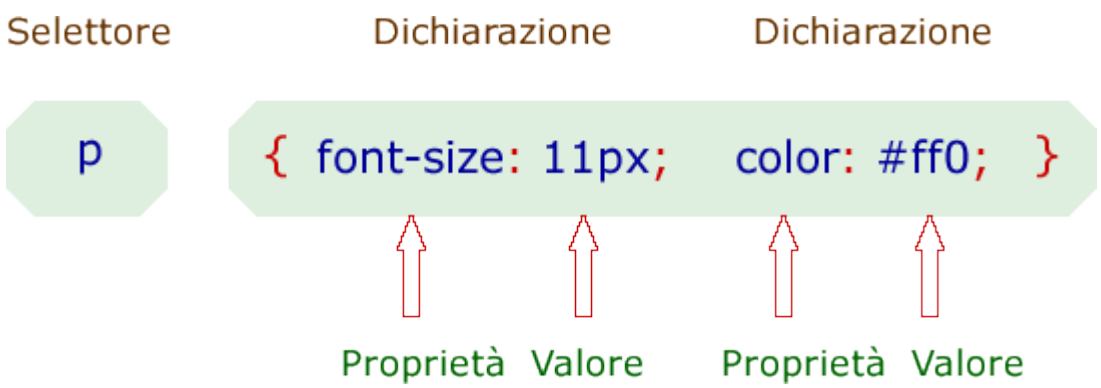


Gli attributi per il Testo

Prima di passare in rassegna le categorie di attributi vorrei ricordare ancora una volta la loro sintassi, a prescindere da quale dei vari modi deciderete di usare.

Abbiamo detto che gli attributi devono essere inseriti all'interno di parentesi graffe { } e laddove in HTML verrebbe usato un "=" (uguale) nei css viene invece usato ":" (due punti).

Argomenti consecutivi sono separati da un ";" (punto e virgola) invece che dalla "," (virgola). Inoltre molti argomenti sono divisi da "-" (trattino centrale) che fa parte integrante del nome stesso.



Gli spazi vuoti ed i ritorno a capo non hanno alcuna influenza.

```
selettore {
  proprietà: valore;
  proprietà: valore;
}
```

Per i valori fare riferimento a questa tabellina.

Vediamo alcuni degli attributi maggiormente adoperati per il testo:

text-align: *left* | *right* | *center* | *justify*

Allineamento del testo, valori possibili: sinistra, destra, centrato, giustificato.

```
p    {text-align: left;}
p    {text-align: right;}
div  {text-align: center;}
div  {text-align: justify;}
```

text-align: left; paragrafo allineato a sinistra.

text-align: right; paragrafo allineato a destra.

text-align: center; paragrafo allineato al centro.

text-align: justify; paragrafo allineato al centro con giustificazione, che cosa è la giustificazione? Guardate attentamente i due margini: destro e sinistro, potrete notare che il testo è allineato sia a destra che a sinistra; per fare questo il comando aggiunge automaticamente degli spazi vuoti fra le parole.

text-decoration: *overline* | *underline* | *line-through* | *blink* | *none*

Decorazione del testo, valori possibili: overline= sopralineato, underline= sottolineato, line-through= sbarrato, none= nessuno. E' anche possibile combinarne più di uno, per esempio inserendo sia underline che overline. None lo si adopera in particolar modo con i link in quanto toglie la sottolineatura che altrimenti avrebbero per default.

```
h1   {text-decoration: overline;}
p    {text-decoration: underline;}
div  {text-decoration: overline underline;}
p    {text-decoration: line-through;}
p    {text-decoration: blink;}
p    {text-decoration: none;}
```

text-decoration: overline; - sopralineato

text-decoration: underline; - sottolineato

text-decoration: overline underline; - sopra e sottolineato

~~text-decoration: line-through; - barrato~~

text-decoration: blink; - lampeggiante (non supportato da IE)

text-decoration: none; - nessuna decorazione.

text-indent: *lunghezza* | *percentuale*

Indentazione, rientro della prima riga di un blocco di testo dal margine sinistro. E' possibile fare uso anche di numeri negativi per un rientro esterno.

```
p     {text-indent: 12px;}
div   {text-indent: 20%;}
```

text-indent: 5px

text-indent: 25px

text-transform: *capitalize* | *uppercase* | *lowercase*

Capitalize trasforma in maiuscola la prima lettera di ogni parola, uppercase e lowercase trasformano in maiuscolo o minuscolo l'interno blocco di testo.

```
P   {text-transform: capitalize;}
P   {text-transform: uppercase;}
P   {text-transform: lowercase;}
```

text-transform: capitalize; testo di prova

TEXT-TRANSFORM: UPPERCASE; TESTO DI PROVA

text-transform: lowercase; testo di prova

line-height: *normal* | *numero* | *percentuale*

Interlinea, altezza della riga del testo, valori possibili: normal, numero, percentuale. Con il valore normal il testo non avrà alcuna variazione. Il numero può essere espresso in em o in percentuale.

```
p    {line-height: 3em;}
div  {line-height: 50%;}
```

line-height: 3em

line-height: 50%

letter-spacing: *numero*

Spazio fra le lettere che compongono il testo. Il numero può essere espresso in tutti i modi possibili come riportato nella tabella parametri.

```
p    {letter-spacing: 1em;}
div  {letter-spacing: 2px;}
```

l e t t e r - s p a c i n g : 1 e m

letter-spacing: 2px

vertical-align: *baseline* | *sub* | *super* | *top* | *middle* | *bottom* | *text-top* | *text-bottom*

Allineamento verticale del testo rispetto ad un riferimento assoluto o all'elemento che lo include.

```
b    {vertical-align: middle;}
span  {vertical-align: top;}
sub  {vertical-align: 60%;}
```

 vertical-align: baseline rispetto alle immagini di lato 

 vertical-align: sub rispetto alle immagini di lato 

 vertical-align: super rispetto alle immagini di lato 

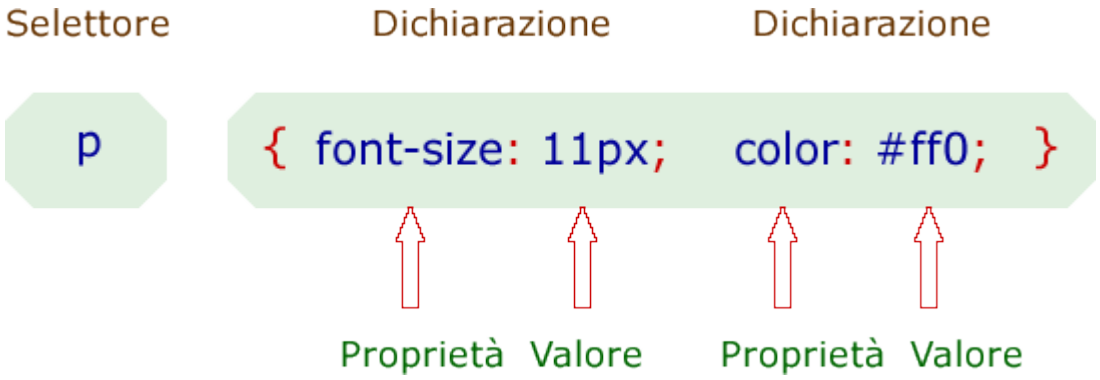
 vertical-align: top rispetto alle immagini di lato 

Gli attributi per il font (carattere)

Prima di passare in rassegna le categorie di attributi vorrei ricordare ancora una volta la loro sintassi, a prescindere da quale dei vari modi deciderete di usare.

Abbiamo detto che gli attributi devono essere inseriti all'interno di parentesi graffe { } e laddove in HTML verrebbe usato un "=" (uguale) nei css viene invece usato ":" (due punti).

Argomenti consecutivi sono separati da un ";" (punto e virgola) invece che dalla "," (virgola). Inoltre molti argomenti sono divisi da "-" (trattino centrale) che fa parte integrante del nome stesso.



Gli spazi vuoti ed i ritorno a capo non hanno alcuna influenza.

```
selettore {
  proprietà: valore;
  proprietà: valore;
}
```

Per i valori fare riferimento a questa tabellina.

Vediamo alcuni degli attributi per il carattere: maggiormente adoperati

font-family: *famiglia1*, *famiglia2*, *famiglia3*;

Si possono elencare più famiglie di caratteri per dare la possibilità al browser di selezionare quella presente sul PC del visitatore. Si tenterà di visualizzare la prima specificata per poi passare a quelle successive nel caso in cui non fossero presenti quelle precedenti. A questo proposito è opportuno chiudere la lista con una famiglia generica come sans-serif per essere certi di trovare un riscontro sempre e comunque.

Le famiglie si separano da una virgola e da uno spazio vuoto opzionale. Nel caso in cui il nome del font fosse composto da più di una sola parola, per esempio *MS Sans Serif*, questo dovrà essere racchiuso fra doppie virgolette "MS Sans Serif" nelle dichiarazioni sul foglio esterno e nella pagina, e da singoli apostrofi nel caso di dichiarazione in linea che farebbe già uso di doppie virgolette per accettare la dichiarazione stessa di style.

```
p   {font-family: verdana, arial, sans-serif;}
div {font-family: "MS Sans Serif", verdana, sans-serif;}


<b style="font-family: 'MS Sans Serif', verdana, sans-serif;"> ... </b>
```

font-family: "MS Serif", "New York", serif;

font-family: verdana, arial, serif;

font-size: *lunghezza* | *valore assoluto* | *valore relativo* | *percentuale*

Si possono esprimere le dimensioni del font facendo riferimento ad uno dei quattro possibili attributi. Vedi anche tabellina parametri.

Di solito si adoperano i punti (pt) o i pixel (px), ma qualsiasi altro parametro andrà sintatticamente bene.

```
p  {font-size: 9pt;}
i  {font-size: larger;}
b  {font-size: 18px;}
h1 {font-size: 150%;}
```

font-size: 9pt;

font-size: larger;

font-size: 18px;

font-size: 150%;

font-style: *normal* | *italic* | *oblique*

Si può eseguire il rendering dei caratteri in corsivo.

```
h1    {font-style: normal;}
p     {font-style: italic;}
b     {font-style: oblique;}
```

font-style: normal;

font-style: italic;

font-style: oblique;

font-variant: *normal* | *small-caps*

Assegna il maiuscoletto con small-caps mentre normal per rimuovere il tutto.

```
p    {font-variant: small-caps;}
b    {font-variant: normal;}
```

FONT-VARIANT: SMALL-CAPS;

font-variant: normal

font-weight: *bold* | *bolder* | *lighter* | *normal* | *100* | *200* | *300* | *400* | *500* | *600* | *700* | *800* | *900*

Per impostare lo spessore del testo. Oltre alle parole riservate è possibile introdurre anche un valore numerico da 100 a 900 con incrementi di 100. Soltanto a partire dal valore 700 si avvertirà l'effetto grassetto.

```
P     {font-weight: bold;}
h1    {font-weight: 700;}
```

font-weight: bold;

font-weight: lighter;

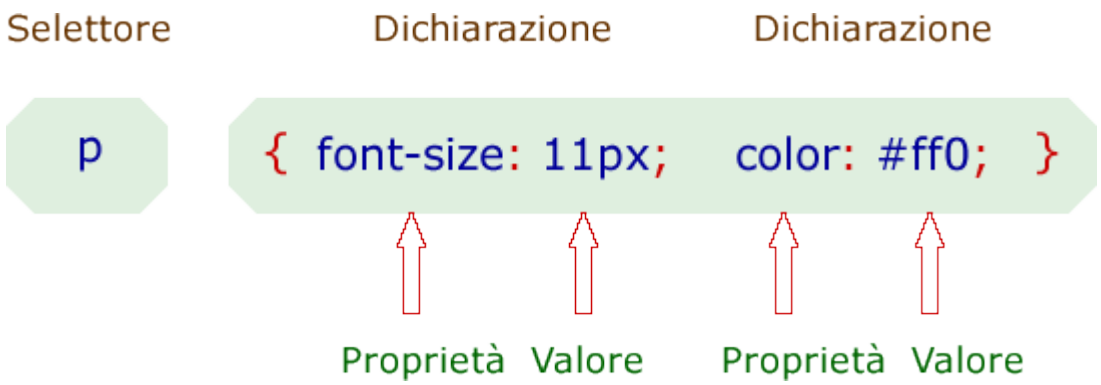
font-weight: 800;

Gli attributi per il per il colore e lo sfondo

Prima di passare in rassegna le categorie di attributi vorrei ricordare ancora una volta la loro sintassi, a prescindere da quale dei vari modi deciderete di usare.

Abbiamo detto che gli attributi devono essere inseriti all'interno di parentesi graffe { } e laddove in HTML verrebbe usato un "=" (uguale) nei css viene invece usato ":" (due punti).

Argomenti consecutivi sono separati da un ";" (punto e virgola) invece che dalla "," (virgola). Inoltre molti argomenti sono divisi da "-" (trattino centrale) che fa parte integrante del nome stesso.



Gli spazi vuoti ed i ritorno a capo non hanno alcuna influenza.

```
selettore {
  proprietà: valore;
  proprietà: valore;
}
```

Per i valori fare riferimento a questa tabellina.

Vediamo alcuni degli attributi per il colore e lo sfondo: maggiormente adoperati

color: *valore*

Questo attributo serve per impostare il colore del testo, fare riferimento alla tabellina per i possibili valori.

```
p     {color: red;}
div   {color: #990099;}
```

```
P {color: red;}
```

```
div {color: #990099;}
```

background-color: valore;

Questo attributo definisce il colore di sfondo di uno style e può essere adoperato indipendentemente dal colore del testo.

Per i possibili parametri fare riferimento alla ormai nota tabellina.

```
p     {background-color: yellow}
div   {background-color: #33ccff}
```

```
P {background-color: yellow;}
continuo a scrivere per meglio
rendere l'idea dello sfondo così applicato.
```

```
DIV {background-color: #33ccff;}
continuo a scrivere per meglio
rendere l'idea dello sfondo così applicato.
```

background-image: url (immagine.gif)

Questo attributo definisce l'immagine di sfondo per un elemento e può essere adoperato indipendentemente dall'immagine e dal colore dello sfondo assegnato alla pagina.

```
p     {background-image: url(file:immagine.gif);}
div   {background-image: url(logo.gif);}
```

p {background-image : url(percorso/immagine.gif);}
continuo a scrivere per meglio
rendere l'idea dello sfondo così applicato.

div {background-image: url (logo.gif);}
continuo a scrivere per meglio
rendere l'idea dello sfondo così applicato.

Se associato al tag body diventa lo sfondo di tutta la pagina.

```
body   {background-image: url("logo.gif");}
```

background-repeat: *repeat* | *repeat-x* | *repeat-y* | *no-repeat*

Questo attributo serve per specificare come disporre l'immagine di sfondo quando non si tratta di uno sfondo omogeneo a tinta unita che riempie tutta la pagina:
repeat indica che l'immagine deve essere replicata in orizzontale ed in verticale.
repeat-x deve essere replicata soltanto in orizzontale.
repeat-y deve essere replicata soltanto in verticale.
no-repeat indica che l'immagine non deve essere replicata.
Logicamente questo attributo dovrà necessariamente essere adoperato in abbinamento con background-image illustrato sopra. Questa l'immagine di sfondo adoperata per gli esempi: 

```
{background-image: url(logo.gif); background-repeat: repeat;}
```

Vedi questo esempio repeat

```
{background-image: url(logo.gif); background-repeat: repeat-x ;}
```

Vedi questo esempio repeat-x

```
{background-image: url(logo.gif); background-repeat: repeat-y;}
```

Vedi questo esempio repeat-y

```
{background-image: url(logo.gif); background-repeat: no-repeat;}
```

Vedi questo esempio no-repeat

background-position: *coordinate*

Questo attributo definisce il punto da cui iniziare a posizionare l'immagine di sfondo e funziona se abbinato con background-image e background-repeat. Si possono adoperare parole chiave per la posizione orizzontale: *left*, *center*, *right* e parole chiave per la posizione verticale: *top*, *center*, *bottom*.

E' anche possibile esprimere valori percentuali, dove: *0% 0%* indica l'angolo in alto a sinistra (default) e: *100% 100%* indica l'angolo in basso a destra. Per maggior precisione si possono specificare anche coordinate espresse in mm, cm, inc.

```
{background-image: url(logo.gif); background-repeat: repeat-y; background-position: 50% 50%;}
```

```
{background-image: url(home.gif); background-repeat: repeat-x; background-position: 50% 50%;}
```

Vedi questi altri esempi:

position 30% 50%

positione left

position center

position right

position top

position top right

position bottom

position bottom left

background-attachment: *scroll* | *fixed*

Questo attributo definisce se l'immagine usata come sfondo deve scorrere sullo schermo insieme al testo (scroll) oppure restare fissa sullo sfondo (fixed) nel momento in cui si scorre la pagina verticalmente.

```
{background-image: url(logo.gif); background-attachment: scroll;}
```

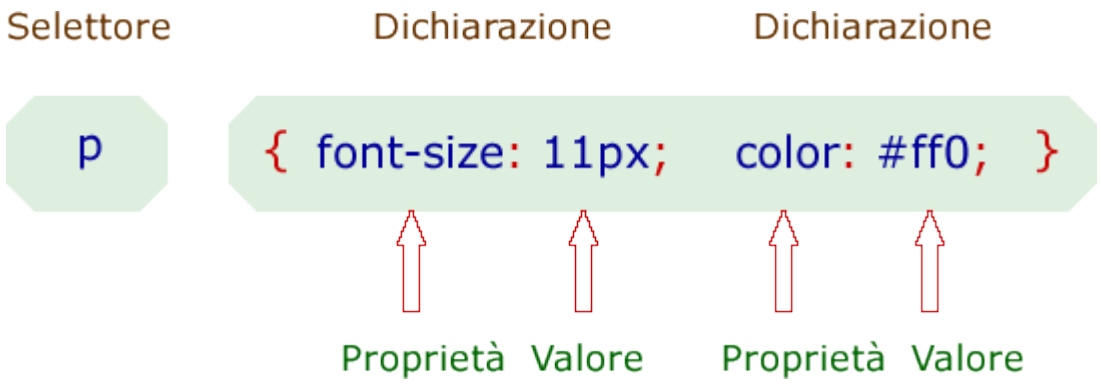
vedi esempio scroll

```
{background-image: url(logo.gif); background-attachment: fixed;}
```

vedi esempio fixed

Gli attributi per il posizionamento dei box

Prima di passare in rassegna le categorie di attributi vorrei ricordare ancora una volta la loro sintassi, a prescindere da quale dei vari modi deciderete di usare.
Abbiamo detto che gli attributi devono essere inseriti all'interno di parentesi graffe { } e laddove in HTML verrebbe usato un "=" (uguale) nei css viene invece usato ":" (due punti).
Argomenti consecutivi sono separati da un ";" (punto e virgola) invece che dalla "," (virgola).
Inoltre molti argomenti sono divisi da "-" (trattino centrale) che fa parte integrante del nome stesso.



Gli spazi vuoti ed i ritorno a capo non hanno alcuna influenza.

```
selettore {
  proprietà: valore;
  proprietà: valore;
}
```

Per i valori fare riferimento a questa tabellina.

Prima di vedere gli attributi di posizionamento è bene ricordare che questi difficilmente saranno adoperati singolarmente, essi infatti fanno parte di un insieme di altri parametri che servono a definire e posizionare un contenitore box all'interno di una pagina web.

Vedere anche quanto detto sul box model a questo proposito, visto che non tutti i browser si comportano allo stesso modo.

position: absolute | *relative* | *fixed* | *static*

Determina il tipo di posizionamento che dovrà assumere il contenitore creato sullo schermo in uno dei possibili modi.

Absolute o Relative, in entrambi i casi l'attributo ha ragione di esistere se abbinato con gli altri attributi di posizionamento.

```
#css {position: absolute; left: 100px; top: 50px;}
```

Contenitore posizionato in modo assoluto a 100px dal margine sinistro e 50px dal margine superiore della finestra del browser.

top: lunghezza | *percentuale* | *auto*

Determina la distanza del contenitore dal margine superiore della finestra del browser. Si possono adoperare tutti i valori ammessi riportati nella tabella parametri.

```
#css {position: absolute; top: 50px; left: 100px;}
```

Contenitore posizionato in modo assoluto a 50px dal bordo superiore della finestra del browser.

left: lunghezza | *percentuale* | *auto*

Determina la distanza del contenitore dal margine laterale sinistro della finestra del browser. Si possono adoperare tutti i valori ammessi riportati nella tabella parametri.

```
#css {position: absolute; top: 50px; left: 100px;}
```

Contenitore posizionato in modo assoluto a 100 px dal margine sinistro della finestra del browser.

width: lunghezza | *percentuale* | *auto*

Determina la larghezza del contenitore se posizionato in modo absolute.

```
#css {position: absolute; width: 300px; left: 100px; top: 50px;}
```

Contenitore con dimensione in larghezza di 300 px posizionato in modo assoluto nella finestra del browser.

height: lunghezza | *percentuale* | *auto*

Determina l'altezza del contenitore se posizionato in modo absolute.

```
#css {position: absolute; height: 400px; ... }
```

Contenitore con dimensione in altezza di 400 px posizionato in modo assoluto nella finestra del browser.

visibility: visible | hidden | collapse
Determina la visibilità del contenitore, questo potrebbe non essere visibile se associato all' attributo hidden.

```
#css {position: absolute; visibility: hidden; height: 400px; ... }
```

Contenitore nascosto. Se non viene specificato alcun attributo per default il contenitore è visible.

z-index: valore

Determina la priorità di visualizzazione del contenitore rispetto allo sfondo e/o rispetto ad altri contenitori posizionati all'interno della stessa pagina.

```
#css {position: absolute; z-index: 10; height: 400px; ... }
```

Contenitore con priorità (10). Il browser IE accetta anche valori negativi che posizionano il contenitore sotto lo sfondo della pagina senza bisogno di assegnare a questa uno z-index superiore.

I Parametri

I Parametri

Sono quei valori che vanno specificati negli attributi e variano a seconda dell'attributo usato. Nelle tabelline riportate sotto è possibile rendersi conto di come possono essere espressi, alcuni di essi sono equivalenti, producono cioè lo stesso risultato, in questo caso sarete voi a decidere quale adoperare in base ai vostri gusti necessità ed abitudini.

Unità di misura:

Ho adoperato l'attributo font-size: *esempio;* con i valori ed i parametri riportati nella colonna risultato.

Unità di misura	Descrizione	Esempio	Risultato
cm	Centimetri	font-size: 1cm;	Risultato
mm	Millimetri	font-size: 7mm;	Risultato

in	Pollici	font-size: 0,25in;	Risultato
pt	Punti	font-size: 12pt;	Risultato
pc	Pica(=12 punti)	font-size: 2pc;	Risultato
px	Pixel	font-size: 25px;	Risultato
em	Lun. lettera m	font-size: 2,5em;	Risultato
ex	Altezza carattere x	font-size: 3ex;	Risultato

Valori assoluti:

Ho adoperato l'attributo font-size: *valore;* con i valori ed i parametri riportati nella colonna risultato.

Valore	Sintassi descrizione	Risultato
xx-small	font-size: xx-small;	Risultato
x-small	font-size: x-small;	Risultato
small	font-size: small;	Risultato
medium	font-size: medium;	Risultato
large	font-size: large;	Risultato

x-large	font-size: x-large;	Risultato
xx-large	font-size: xx-large;	Risultato

Valori relativi:

Ho adoperato l'attributo font-size: *valore;* con i valori ed i parametri riportati nella colonna risultato.

Valore	Sintassi descrizione	Risultato
smaller	font-size: smaller;	Risultato
larger	font-size: larger;	Risultato

Colori:

Ho adoperato l'attributo color: *valore;* con i valori ed i parametri riportati nella colonna risultato.

Valore	Sintassi descrizione	Risultato
nome	color: green	Risultato
valore esadecimale	color: #008000;	Risultato
valore RGB	color: rgb(0,128,0);	Risultato
percentuale RGB	color: rgb(0%,50%,0%);	Risultato

Posizionamento Box

Attributi e Parametri per definire e posizionare un box, esempio di riferimento:

```
<div style="position: absolute; left: 100px; top: 50px; z-indez: 5;"> contenuto </div>
```

con valori e parametri riportati nella tabella sotto.

Attributo	Parametro	Descrizione
position	*absolute* - *static* - *relative*	Il modo in cui può essere posizionato un box contenitore.

left	*cm* - *mm* - *in* - *pt* - *pc* - *px* - *ex* - *em*	la distanza del contenitore dal margine sinistro della finestra del browser.
top	*cm* - *mm* - *in* - *pt* - *pc* - *px* - *ex* - *em*	la distanza del contenitore dal margine superiore della finestra del browser.
height	*cm* - *mm* - *in* - *pt* - *pc* - *px* - *ex* - *em*	Altezza del contenitore se posizionato in modo absolute.
width	*cm* - *mm* - *in* - *pt* - *pc* - *px* - *ex* - *em*	Larghezza del contenitore se posizionato in modo absolute.
overflow	*visible* - *hidden* - *scroll*	Contenuto che supera Larghezza Altezza del box contenitore.
visibility	*visible* - *hidden* -	Visibilità o meno di un box contenitore.
Z-index	*xx* (numeri positivi)	L'ordine di priorità di visualizzazione di un box contenitore.

Foglio di style di Stampa

Abbiamo visto come definire i vari styli e questi sono interpretati dal browser che li esegue a video, è anche possibile personalizzare un foglio di style specifico per la sola stampa in modo da avere margini, font, interruzioni pagina, nascondere parti che non devono essere stampate ed altro ancora.

Per farlo è sufficiente creare un secondo foglio di style al quale assegnare come attributo del parametro media *print* al posto di *screen*

```
<link rel=stylesheet href="nome_stampa.css" type="text/css" media="print">
```

Così facendo i due fogli di style (screen e print) saranno autonomi e ben distinti fra di loro, Se non vogliamo creare un foglio di style a parte per la sola stampa è possibile definire i parametri anche all'interno dello stesso foglio facendo uso della regola @media che permette di raggruppare e suddividere le varie definizioni a seconda del media al quale si riferiscono.

```
@media screen {
body {  margin-left: .5cm;
        margin-right: .5cm;
        color: #000099;
        font-family: Verdana, Arial, sans-serif;
        font-size: 10pt;
        }
}

@media print{
body {  margin-left: 2px;
        margin-right: 2px;
        color: #000000;
        font-family: Verdana, Arial, sans-serif;
        font-size: 12pt;
        }
}
```

Così facendo avremo parametri differenti a seconda che lo stesso documento sia visto a schermo o stampato su carta, ne consegue che ognuno di potrà fare tutte le personalizzazioni che desidera, tenuto conto anche del fatto che potrà scegliere cosa vedere e cosa stampare semplicemente assegnando ai blocchi interessati una classe, un ID, o ridefinendo il blocco nel quale sarà stato impostato display: none; tanto per fare un esempio.

Anche in questo caso ci sono differenze fra browser e browser ed alcuni non gestiscono allo stesso modo la stampa di blocchi posizionati in modo assoluto, pazienza, con qualche prova riuscirete ad ottenere stampe più che accettabili.

Personalmente preferisco avere due fogli esterni ben distinti come riportato nell'esempio sotto

```
<link rel=stylesheet href="foglio_video.css" type="text/css" media="screen">
<link rel=stylesheet href="foglio_stampa.css" type="text/css" media="print">
```

sui quali andare di volta in volta ad intervenire, questo però dipende solo dalla propria organizzazione mentale ed ognuno usi il metodo che ritiene più comodo.

Trattandosi di stampe si ha a che fare con il salto pagina o interruzione di stampa, per evitare di spezzare il documento a fine pagina in una parte di testo importante, è possibile decidere quando attivare una interruzione di stampa grazie alle proprietà: page-break-before e page-break-after (prima e dopo) che accettano valori quali: *always*, *avoid*, *left*, *right* e *auto*.

Alcuni non sono supportati da tutti i browser, altri sono di scarso utilizzo, alla fine quelli che vengono realmente adoperati sono soltanto i primi due: *always* e *avoid* rispettivamente per inviare una interruzione di pagina o per impedire una interruzione di pagina al blocco al quale vengono applicati.

```
div.inizio {page-break-before: always;}
div.fine {page-break-after: always;}
```

Con questo esempio il browser inserisce un'interruzione di pagina prima del blocco * * ed una dopo il blocco * *

Conclusioni: avrei finito, consapevole che ci sarebbe ancora molto da dire, lascio questo compito alla letteratura specializzata o a quei siti che si occupano soltanto di questo argomento.

Per quanto mi riguarda considero questa guida più che sufficiente per poter apprendere e capire le varie tecniche di programmazione per questo sistema che si applica al buon "vecchio" HTML.

Non mi resta che salutarvi e magari con un esempio di scritta gigante, in html le dimensioni massime ottenibili dall'elemento font sono

Queste

e cioè size 7. Con i CSS non ci sono limiti se non quelli dello schermo...

Ecco una dimostrazione

Vi saluto con un grosso ciao... lo so che poteva essere anche più grande, ma accontentatevi :)

CIAO

Vi ricordo due indirizzi sui quali reperire ulteriori informazioni: specifiche del W3C, dove troverete tutti gli attributi ed i parametri ammessi e supportati dai CSS1 e: specifiche per i CSS2. Ormai prossimi anche i CSS3 con nuovi elementi rivoluzionari ma al momento non ancora supportati da tutti i browser.