

# Cascading Style Sheets

Claudio Cicali



Creative Commons License

**Attribution-NonCommercial-ShareAlike 2.5**

<http://creativecommons.org/licenses/by-nc-sa/2.5/>

[claudio@cicali.org](mailto:claudio@cicali.org)

[http://claudio.cicali.org/files/cicali\\_css.pdf](http://claudio.cicali.org/files/cicali_css.pdf)

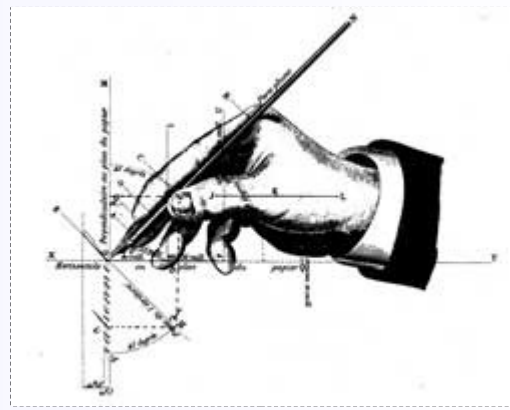
Web design, ovvero:

- **Struttura**
- **Presentazione**
- **Comportamento**

Web design, ovvero:

- **Struttura** ✓ **(X)HTML**
- **Presentazione** ✓ **CSS**
- **Comportamento** ✓ **DOM/Ecmascript**

# Introduzione



Dall'introduzione del W3C:

*CSS è un linguaggio di stile che permette ad **autori** ed **utenti** di associare uno stile (per esempio gestione dei font, dei colori, delle spaziature) a documenti strutturati (come documenti HTML o applicazioni XML).*

*Separando lo stile di presentazione dal contenuto dei documenti, CSS semplifica la pubblicazione dei documenti Web e la manutenzione dei siti.*

**E' importante notare che gli attori sono dunque due:**

L'autore

L'utente

# Il concetto

```
<h1>Introduzione ai CSS</h1>
```

+

“Gli h1 devono essere visualizzati in rosso e in corsivo”

=

***Introduzione ai css***

# Standard?

I css sono uno **standard W3C**, per cui sono raccomandazioni

Come per HTML, nel tempo si sono avute molte "personalizzazioni"

## Mozilla

```
-moz-border-radius-top-left  
-moz-border-radius-bottom-left  
-moz-border-radius-top-right  
-moz-border-radius-bottom-right
```

## Internet Explorer

```
scrollbar-3dlight-color  
scrollbar-arrow-color  
scrollbar-base-color  
scrollbar-darkshadow-color  
scrollbar-face-color  
scrollbar-highlight-color  
scrollbar-shadow-color
```

# Stato dello standard

## **CSS 1**

*Recommendation* dal 17 dicembre 1996  
*Riviste* l'11 gennaio 1999

## **CSS 2**

*Recommendation* dal 12 maggio 1998

## **CSS 2.1**

*"Last call" Working draft* dal luglio 2005

## **CSS 3**

Modulare

Work in progress

# Le differenze tra le versioni

## **CSS 2**

- introduce il concetto di @media
- posizionamento degli elementi

## **CSS 2.1**

- corregge degli errori delle specifiche 2.0
- inserisce qualcosa delle specifiche 3.0 ormai già implementate
- "snapshot" dell'uso dei CSS al momento della sua stesura

## **CSS 3**

- E' un sistema modulare (~31 moduli)

# Basi del linguaggio

I css si scrivono usando

REGOLE

COMMENTI

REGOLE @ (“at rules”)

# Le regole

Ogni regola è divisa tra

**SELETTORE** e **DICHIARAZIONE**

come per dire:

**Questo** deve apparire **così**

# Le regole

## Com'è fatta una regola

SELETTORE DICHIARAZIONE



```
PROPRIETA': VALORE/I;  
PROPRIETA': VALORE/I;  
PROPRIETA': VALORE/I;
```

...

# Le dichiarazioni

```
color: red;  
font-style: italic;  
margin-top: 10px;
```

**color, font-style, margin-top** sono proprietà

**red, italic, 10px** sono valori

# Ma questi CSS...

**Come si usano?**

**Dove si mettono?**

**Come si associano gli stili all'HTML?**

# Modalità **INLINE**

Si fa uso dell'attributo **STYLE**;

Il selettore è "implicito" in quanto la dichiarazione si riferisce direttamente all'elemento di cui l'attributo fa parte

```
<p style="background-color: yellow">  
  Testo con lo sfondo giallo  
</p>
```

```
<p style="background-color: yellow; font-size: 10px">  
  Testo con lo sfondo giallo e con font di 10 pixel  
</p>
```

# Esercizio

Scrivere il proprio nome  
colorando ogni singola lettera

C L A U D I O C I C A L I

Proprietà da usare:

**color**

**font-size**

**font-weight**

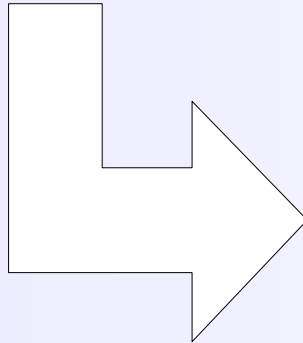
**letter-spacing**

# Utilizzo dei selettori

SELETTORE {

DICHIARAZIONE

}



SELETTORE {

proprietà: valore;  
proprietà: valore;  
proprietà: valore;

}

Anche: SELETTORE { DICHIARAZIONE }

# Utilizzo dei selettori

Esistono diversi TIPI di selettore

Dunque, esistono diverse modalità di selezionare una parte del documento

Una delle difficoltà dei CSS sta nel capire quale sia la modalità migliore in un determinato contesto

# Tipi di selettore

## Type selector

Selezionano gli elementi HTML  
tramite il loro nome

```
p { color: #fff; }  
em { font-style: italic; }  
body { margin: 0; padding: 0; }
```

# Modalità **EMBEDDED**

```
<html>
<head>
<title>Esempio</title>
<style type="text/css">
  /* Un commento */
  body {
    background-color: white;
    margin: 0;
    padding: 0;
  }

  /* Un commento su
     più righe */
  h1 {font-size: 2em;}
</style>
</head>
<body>
  ...
```

Si fa uso dell'elemento  
**STYLE del HEAD;**

Si possono aggiungere commenti

# Modalità **EXTERNAL**

Si usa un riferimento, tramite l'elemento **link**, ad un file esterno

```
<html>
<head>
<title>Esempio</title>
<link rel="stylesheet" type="text/css" href="style/miostile.css">
</head>
<body>
...
```

# Come si usano, RIEPILOGO

## External

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

## Embedded

```
<style type="text/css">  
  body {background-color: white;}  
</style>
```

## Inline

```
<p style="text-align: justify">
```

# Tipi di selettore

## Class selector

Si usa l'attributo **CLASS** (introdotto con HTML4)

Seleziona tutti gli elementi HTML il cui attributo **class** contiene il valore specificato

Può essere unito ad un **type selector**

La classe viene specificata tramite il punto "."

```
<p class="header">  
<div class="news-body">  
<h1 class="header">
```

```
.header { margin: 10px; padding: 0; }  
p.header {color: #fff;}  
div.news-body {color: #fff;}
```

# Tipi di selettore

## Class selector

L'attributo CLASS può specificare l'appartenenza a **più classi**

In tal caso le dichiarazioni vengono "sommate"

```
<li class="menu-item first">  
<li class="menu-item">  
<li class="menu-item">  
<li class="menu-item last">
```

```
li.menu-item {  
    font-size: 18px;  
    background-color: red;  
}  
  
li.first {  
    color: pink;  
    background-color: yellow;  
}
```

# Tipi di selettore

## ID selector

Seleziona l'elemento che ha un particolare ID

L'ID viene specificato tramite il carattere “#”

Si usa l'attributo ID (introdotto in HTML 4)

```
<div id="header">
```

```
div#header { margin: 10px; padding: 0;}  
p#titolo {font-weight: bold;}
```

# ID Selector - 2

Gli ID non devono ripetersi!

```
<div id="header">  
  ...  
</div>  
  
<p id="header">  
  ...  
</p>
```

# ID e CLASS

Classe e ID possono convivere  
felicemente!

```
<div id="left_menu" class="big_box">  
  ...
```

# Regole di igiene

- Non usare nomi di classi (tantomeno ID) che specifichino l'apparenza di un elemento ("red", "blue", "yellow\_box")
- Usare gli id solo per suddividere semanticamente sezioni del documento
- Usare le classi per aggiungere semantica a particolari elementi
- Usare le classi con parsimonia, privilegiando gli elementi HTML (vedi anche il *descendant selector*)

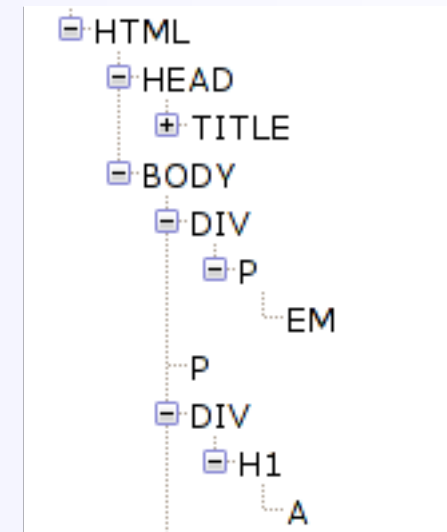
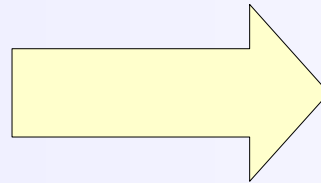
Sui nomi non esiste una regola precisa.

Conviene chiedersi:

*"Se fosse un tag HTML, sarebbe deprecato o no?"*

# Padri, figli e discendenti

```
<html>
<head>
  <title></title>
</head>
<body>
  <div>
    <p>
      <em></em>
    </p>
  </div>
  <p></p>
  <div>
    <h1>
      <a></a>
    </h1>
  </div>
</body>
</html>
```



# Ereditarietà

E' un concetto abbastanza intuitivo  
ma di fondamentale importanza

Gli elementi possono ereditare il valore  
di determinate proprietà del loro "antenato"

Non tutte le proprietà vengono ereditate:  
**border** per esempio non è ereditata.

Esiste anche l'apposito valore keyword **inherit**

# Tipi di selettore

## Descendant selector

Seleziona un elemento "discendente" specificando la gerarchia di "parentela" che esso ha con i suoi predecessori

Si definisce elencando la gerarchia degli elementi separando ogni termine con uno spazio.

```
p.question strong { font-weight: 110%;}  
div#header div.header_body h1 {font-family: Arial;}
```

```
<p class="question">  
  <strong>Prima domanda</strong>  
  Testo della domanda  
</p>
```

```
<div id="header">  
  <div id="header_body">  
    <div>  
      <h1>title</h1>  
    </div>  
  </div>  
</div>
```

# Raggruppamento dei selettori

I selettori si possono raggruppare,  
al fine di scrivere meno codice:  
**MENO CODICE, MENO ERRORI**

```
h1 {  
  margin-top: 1em;  
}  
  
h2#intro {  
  margin-top: 1em;  
}  
  
p.heading {  
  margin-top: 1em;  
  color: navy;  
}
```

Equivale a



Si raggruppano con la virgola:

```
h1, h2#intro, p.heading {  
  margin-top: 1em;  
}  
  
p.heading {  
  color: navy;  
}
```

# Il selettore universale

\*

# Il selettore universale

```
* {color: cyan;}  
  
body * UL {color: gray;}  
  
DIV.danger * {color: red;}
```

# Tipi di selettore

## Link pseudo class selector

Sono utilizzati per selezionare i diversi stati nei quali si può trovare l'elemento "a" o una sua classe o id. Si definiscono con un "a:stato"  
Gli stati intercettabili sono:

a:link (stato "normale")  
a:visited (stato "visitato")  
a:active (stato "lo sto cliccando")

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

# Link pseudo class selector - 2

A causa delle precise regole di *specificità* dei CSS, gli pseudo class selector devono essere definiti nel seguente preciso ordine:

```
a{}  
  
a:link {}  
  
a:visited {}  
  
a:hover {}  
  
a:active {}
```

**LoVe and HAte**

# Tipi di selettore

## Dynamic pseudo class selector

Sono utilizzati per selezionare i diversi stati nei quali si può trovare un elemento. Si selezionano con "elemento:stato"

Gli stati selezionabili sono:

:hover (stato "mouse over")  
:focus (l'elemento ha il focus)

```
a:hover { text-decoration: none; }  
form.anagrafica input:focus { background: ivory; }
```

IE 6 non supporta **:focus** e supporta **:hover** solo sugli anchor

# Tipi di selettore (CSS 2.x)

## Pseudo element selector

```
p:first-line {}  
p:first-letter {}  
p:before { content: "Test" }  
p:after { content: "Test" }
```

## Child selector

```
div>strong {}
```

## First child selector

```
p:first-child {}
```

## Adjacent sibling selector

```
strong + em {}
```

## Attribute selector

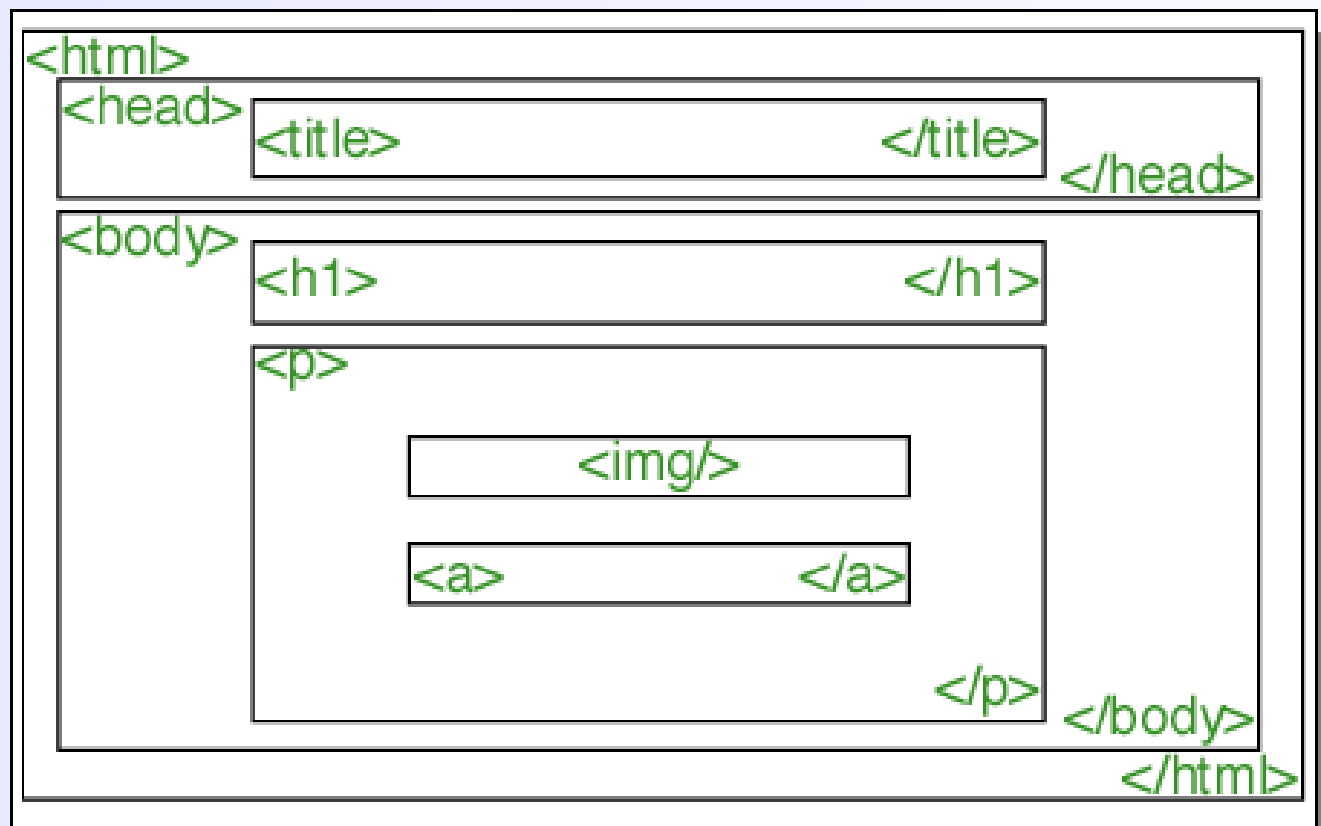
```
a[title="menu principale"] {}  
input[type="password"] {}
```

Nessuno di questi selettori è supportato da **IE6**

# Box Model

Per ogni elemento, inline o block, il browser crea intorno ad esso un **BOX**

Ognuno di questi box può contenere altri box  
In tal caso diventa un *containing box*



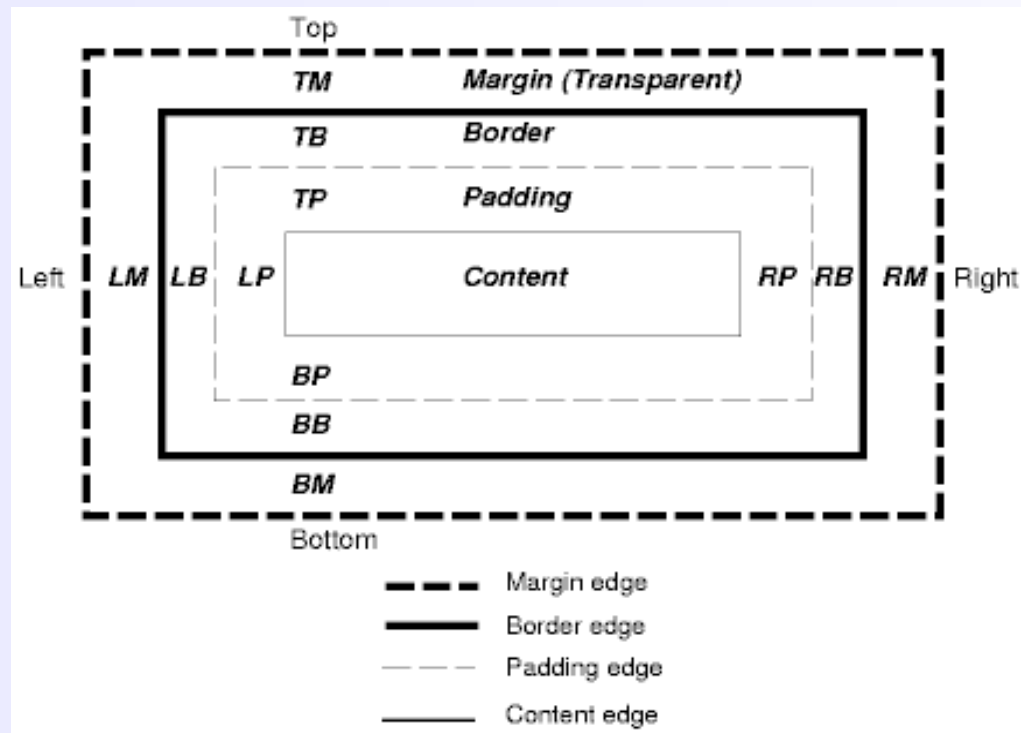
# Box Model

Il *box model* specifica le regole per calcolare la larghezza e l'altezza *effettiva* di questi box generati, il loro *ingombro*.

Ogni box ha:

- un **margin** che lo separa dagli altri elementi
- un **border**, ovvero una cornice con un certo spessore
- un **padding**, ovvero uno spazio trasparente che separa il contenuto dal bordo

# Box Model



“All things being equal, a box's width depends on the width of its parent, but its height depends on the height of its children.”

# Box Model

Quando si parla di *larghezza* (width) si parla sempre di larghezza del contenuto del box

Tale larghezza può essere dichiarata esplicitamente tramite la proprietà **width** ma solo per gli elementi block level (non per gli inline)

# Box Model

La larghezza totale (l'ingombro fisico orizzontale) di un box è data dalla somma di:

- larghezza del contenuto
- quantità di margine sinistro destro
- spessore del bordo sinistro e destro
- quantità di padding sinistro e destro

In modo del simile si calcola l'effettiva *altezza del box*

# Box Model

## Collassamento dei margini - 2

```
<div id="masthead">  
<h1>ConHugeCo</h1>  
<p>Making the world safe for super sizes</p>  
</div>
```

```
#masthead {background: #F80; margin: 10px;}  
#masthead h1 {margin: 20px 10px;}  
#masthead p {font-style: italic;}
```

**ConHugeCo**

*Making the world safe for super sizes*

← Come appare

Margini rivelati →

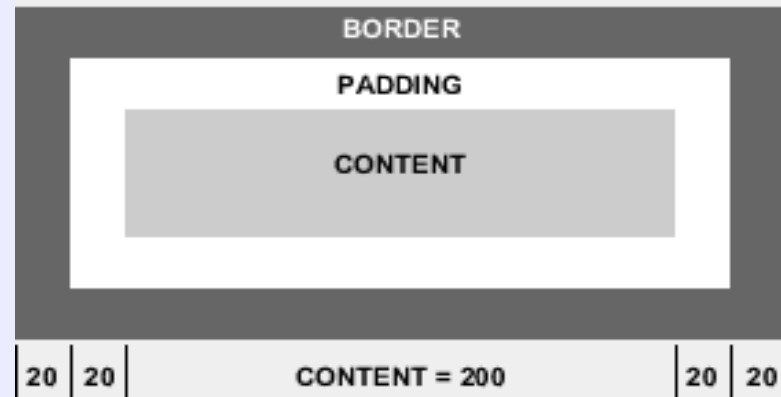
**ConHugeCo**

*Making the world safe for super sizes*

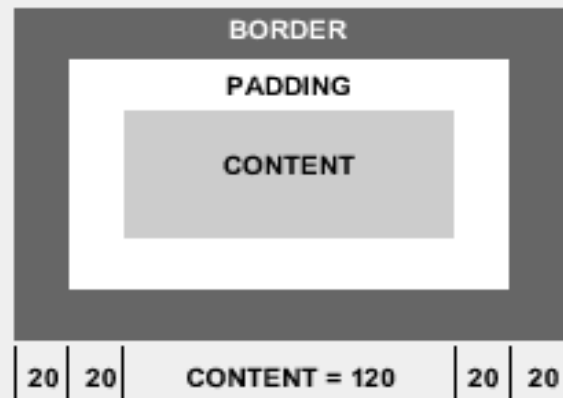
# Box model, IE 5.x e gli altri

```
{ width: 200px; padding: 20px; border: 20px; }
```

**CORRECT BOX MODEL - TOTAL WIDTH = 280**



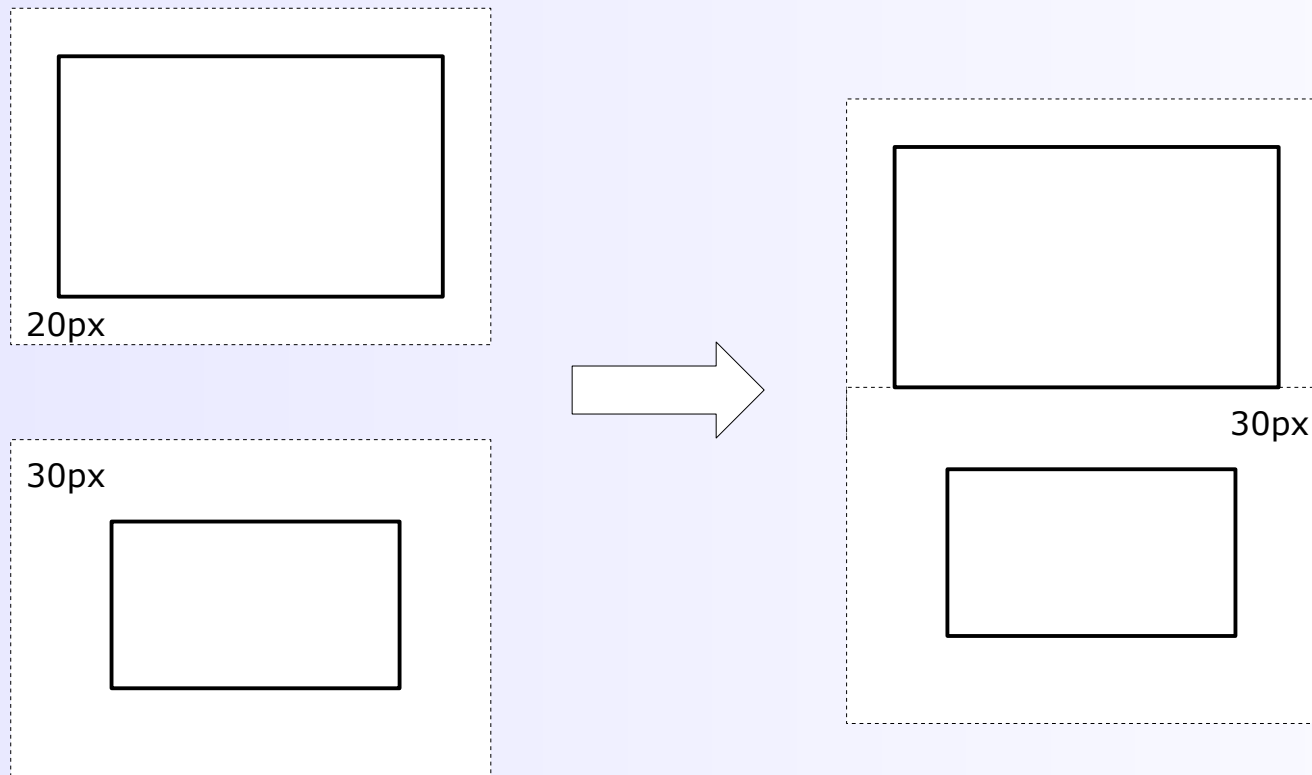
**IE BOX MODEL - TOTAL WIDTH = 200**



# Box Model

I margini **verticali** *collassano*: il margine più piccolo viene assorbito

$$2 + 2 = 2$$



# Box Model

## Collassamento dei margini - 3

Per evitare l'effetto *collapse* precedente:

- si imposta un border o un padding al container (masthead)
- oppure si toglie il margin a h1 e p e si usa il padding

Risultato finale:



# Tipi di valore

I valori delle proprietà CSS possono essere espressi tramite i seguenti tipi di valore:

- Parole chiave (*keywords*)
- Colori
- Liste
- Percentuali
- URI
- Grandezze espresse in unità di misura

# Tipi di valore – Parole chiave











```
bold, top, left, bottom, no-repeat, repeat-x  
large, medium, small, collapse, separated...
```

# Tipi di valore - Colori

Si esprimono tramite una tripletta di valori esadecimali preceduta dal carattere "#". Per molti colori è possibile usare un nome mnemonico.

Gli elementi della tripletta indicano la quantità della singola componente RGB

Può essere usata anche la parola chiave **rgb**

Named	Numeric	Color name	Hex rgb	Decimal
		aliceblue	#F0F8FF	240,248,255
		antiquewhite	#FAEBD7	250,235,215
		aqua	#00FFFF	0,255,255
		aquamarine	#7FFFD4	127,255,212
		azure	#F0FFFF	240,255,255

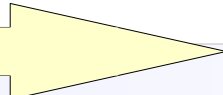
<http://www.w3.org/TR/2003/CR-css3-color-20030514/>

```
p { color: red; }  
p { color: #ff0000; }  
p { color: #f00; }  
p { color: rgb(255, 0, 0); }
```

# Tipi di valore - Liste

Viene usata una lista quando si devono specificare i font da usare per un particolare elemento

```
p.intro {  
    font-family: Verdana, Helvetica, sans-serif;  
}
```

Ordine di preferenza 

Le famiglie specificabili sono 5:

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

# Tipi di valore - Percentuali

Le percentuali sono numeri sempre positivi, ma anche maggiori di 100

```
body {  
    font-size: 76%;  
}  
  
h1 {  
    font-size: 140%;  
}  
  
div.box {  
    width: 15%;  
}
```

Viene sempre ereditato il valore *calcolato*

# Tipi di valore - URL

Si usano quando sia necessario specificare da CSS una risorsa (immagine o file):

```
@import url(elements.css);  
  
div.header {  
    background-image: url(logo.gif);  
}
```

# Tipi di valore - Grandezze

## Grandezze assolute:

- **pt**  
point, 1/72 di pollice
- **in** (2,54cm), **cm**, **mm**
- **pc:**  
pica (1 pica = 12 pt)

## Grandezze relative:

- **px**  
pixel del media attuale (dipende dalla risoluzione)
- **em**  
The 'em' unit is equal to the computed value of the 'font-size' property of the element on which it is used. The exception is when 'em' occurs in the value of the 'font-size' property itself, in which case it refers to the font size of the parent element.
- **ex**  
da moltiplicare per l'x-height dell'elemento (altezza della "x")

# The winner is...

Usare **em** è cosa buona e giusta  
perché  
è un valore relativo ad una  
precisa scelte dell'utente

Ma...

# IE, ancora una volta

- La dimensione del font, di default, è 16px
- Troppo grande. Meglio imporre 0.8em al BODY
- Ma questo rende illeggibile "smaller font" su IE
- Occorre impostare dunque la grandezza del font del body **in percentuale**, circa al 80% (76%)

Il motivo?

*"There is no apparent reason for this to work, but it does. "*

<http://www.alistapart.com/articles/elastic/>

# Visual formatting model

Interazioni, posizionamento e visualizzazione dei box

Tre schemi di posizionamento:

**Normal Flow**

**Absolute**

**Float**

## Apparizioni e sparizioni: la proprietà **display**

Si distinguono gli elementi "block" e "inline" ovvero, appunto, gli elementi che hanno il valore "block" e "inline" della proprietà **display**

Nel *flusso* gli elementi "block" sono disposti verticalmente uno sull'altro, mentre gli inline orizzontalmente, uno accanto all'altro.

Qualsiasi elemento può essere trasformato in block o inline a dispetto del suo default, tramite la proprietà CSS "display" :

- display: block
- display: inline

Un elemento può anche essere tolto completamente dal flusso impostando il valore della proprietà a "none"

## Il posizionamento degli elementi

E' possibile agire sull'effettivo posizionamento degli elementi tramite la proprietà **position** e le proprietà **top, bottom, left, right**

Nel *normal flow* sono previsti i valori **static e relative**

La position **absolute** toglie gli elementi dal flusso e li ridispone ovunque

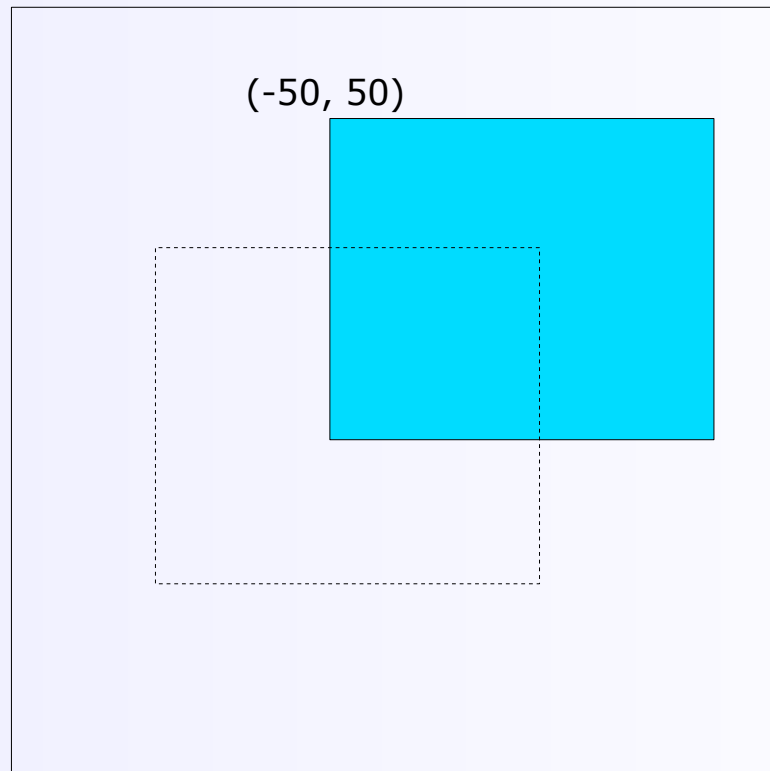
Per lo schema di posizionamento **float** si usa invece la proprietà **float** e i possibili valori **left, right, none**

# Visual Formatting Model

## position: relative

Un elemento per la quale sia definita una posizione **relative**, viene inserito nel normale flusso e poi spostato secondo gli offset specificati, rispetto alla sua teorica posizione.

```
#outer {  
  position: relative;  
  top: -50px;  
  left: 50px;  
  width: 100px;  
  height: 100px;  
}
```

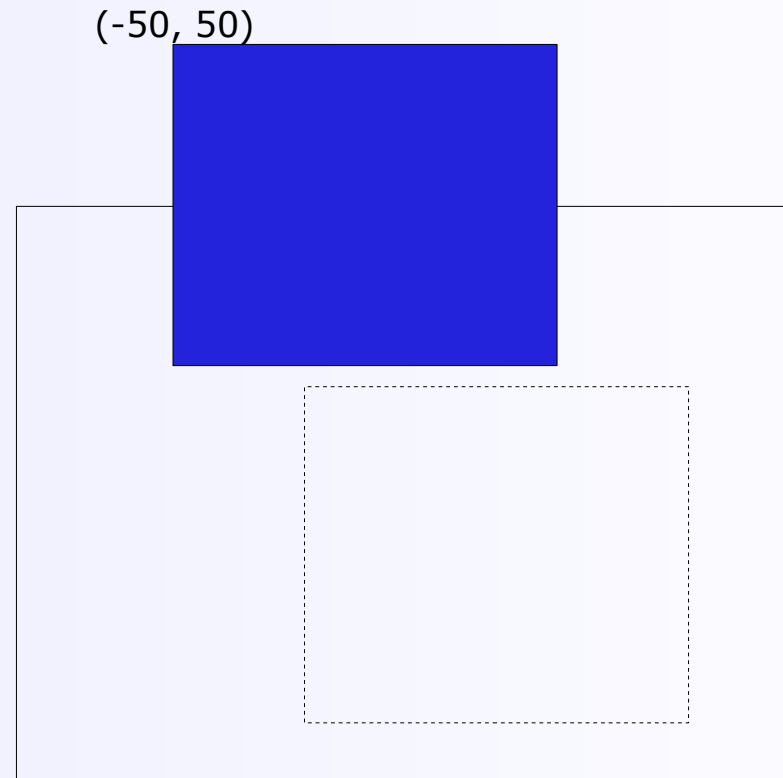


# Visual Formatting Model

## position: absolute

Un elemento per il quale sia definita una posizione assoluta, viene tolto dal flusso e posizionato direttamente, secondo gli offset specificati, rispetto al suo "contenitore" (che deve essere, a sua volta, un box posizionato con "position") o al "body".

```
#outer {  
  position: absolute;  
  top: -50px;  
  left: 50px;  
  width: 100px;  
  height: 100px;  
}
```



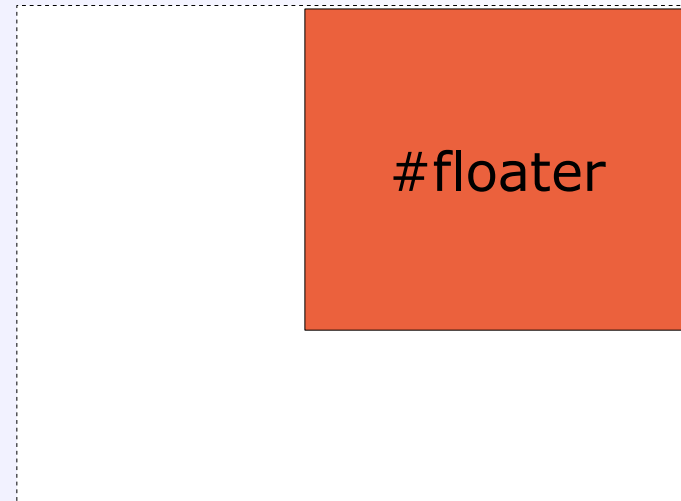
# Visual Formatting Model

## float: left/right

Un elemento per il quale sia definita la proprietà **float** verrà tolto dal normale flusso e sarà posizionato nel punto più a sinistra (o a destra) possibile rispetto al suo contenitore.

Qualsiasi elemento flottante divente di tipo "block" e sarà dunque possibile imporre ad esso una larghezza esplicita

```
#floater {  
  float: right;  
  width: 100px;  
  height: 100px;  
}
```



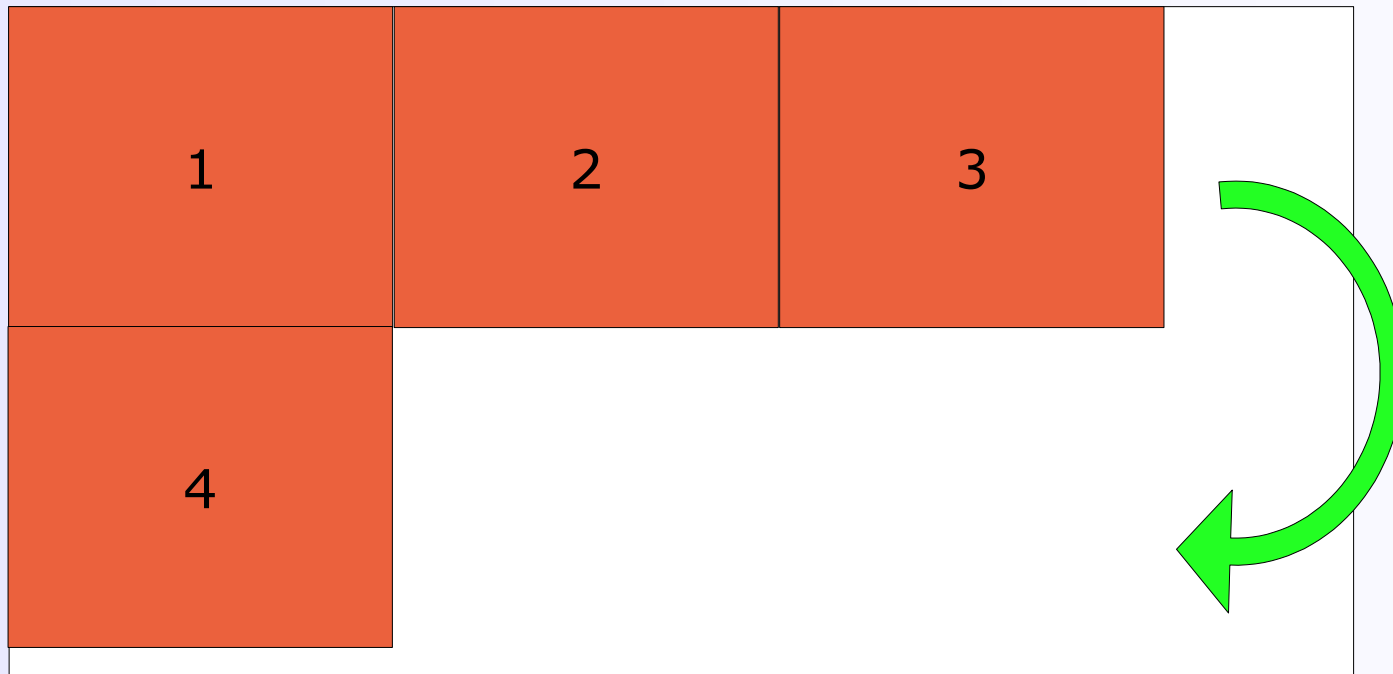
# Visual Formatting Model

## float

```
#container {  
  width: 350px;  
  height: 230px;  
}  
.floater {  
  float: left;  
  width: 100px;  
  height: 100px;  
}
```

Blocchi flottanti contigui *will wrap*

```
<div id="container">  
  <div class="floater">1</div>  
  <div class="floater">2</div>  
  <div class="floater">3</div>  
  <div class="floater">4</div>  
</div>
```



# Visual Formatting Model

## float: left/right (2)

*Content displacing*: una caratteristica dei blocchi *float* è che qualsiasi contenuto *inline* che dovesse seguire verrà posizionato intorno ad essi, sulla destra o sulla sinistra.

```
#floater {  
  float: left;  
}
```

```
  
<p>  
blah blah blah blah blah blah blah blah  
...  
</p>
```



blah blah blah  
blah blah blah  
blah blah blah  
blah blah blah  
blah blah blah  
blah blah blah

# Proprietà **clear**

E' possibile disabilitare l'effetto *displacing* utilizzando la proprietà **clear** sull'elemento che segue un float

```
#floater {  
  float: left;  
}  
  
p {  
  clear: left;  
}
```

```
  
<p>  
blah blah blah blah blah blah blah blah  
...  
</p>
```

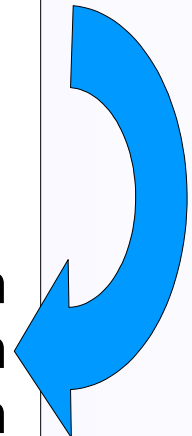
I valori della proprietà sono:

- **left**
- **right**
- **both**



blah blah blah  
blah blah blah  
blah blah blah  
blah blah blah

blah blah blah blah blah blah  
blah blah blah blah blah blah  
blah blah blah blah blah blah  
blah blah blah



# Position: fixed

Blocca fisicamente la posizione di un elemento direttamente sulla viewport

Ha apparentemente lo stesso comportamento della position absolute

Poco utile, in quanto non supportato da IE

# Origine

I CSS possono arrivare da tre fonti (origine).

In ordine di priorità:

- **CSS Autore**
- **CSS Utente**
- **CSS Predefiniti (UA)**

# Peso

I CSS utente hanno normalmente priorità massima (peso)

Il peso di una regola CSS può essere invertito tramite la parola chiave **!important**

Autore:

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

>

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

<

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

<

Utente:

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

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

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

(L'ultimo esempio era invertito nei CSS1)

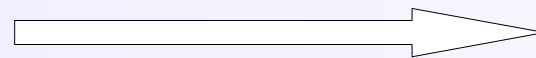
# Specificità

Ci dà (e dà al browser) una misura quanto più una regola sia *precisa* rispetto alle altre

Si usa la regola della tripletta:

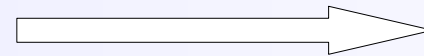
- ID
- CLASSI (e :pseudoclassi)
- ELEMENTI

```
p { color: yellow }
```



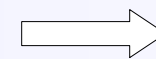
0 - 0 - 1

```
#header p { color: red }
```



1 - 0 - 1

```
#header p.warning { color: blue }
```



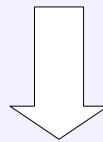
1 - 1 - 1

# Specificità: esempio

```
h2 {color: red;}  
.titolo {color: green;}  
#titolo2 {color:blue;}
```

+

```
<h2 class="titolo" id="titolo2">Titolo di esempio</h2>
```



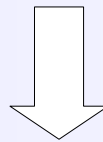
**Titolo di esempio**

# Specificità: esempio

```
h2 {color: red;}  
.titolo {color: green;}  
#titolo2 {color:blue;}
```

+

```
<h2 class="titolo" id="titolo2">Titolo di esempio</h2>
```



**Titolo di esempio**

# Video, stampanti e altro

I CSS permettono di specificare come un documento dovrà APPARIRE, ma è possibile anche definire un diverso aspetto a secondo dell'utilizzo (media) che viene fatto del documento

E' possibile infatti specificare come un certo documento debba apparire in fase di stampa... o in braille... o sintesi vocale...

# Media

- **all** Il CSS si applica a tutti i dispositivi di visualizzazione.
- **screen** Valore usato per la resa sui normali browser web.
- **print** Il CSS viene applicato in fase di stampa del documento.
- **projection** Usato per presentazioni e proiezioni a tutto schermo.
- **aural** Da usare per dispositivi come browser a sintesi vocale.
- **braille** Il CSS viene usato per supporti basati sull'uso del braille.
- **embossed** Per stampanti braille.
- **handheld** Palmari e simili.
- **tty** Dispositivi a carattere fisso.
- **tv** Web-tv

<http://www.codestyle.org/css/media/>

# Specificare il media

## Attributo MEDIA dell'elemento LINK

```
<link rel="stylesheet" media="all" href="default.css" type...>
```

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

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

```
<link rel="stylesheet" media="print, screen" href="miostile.css" ...>
```

# Specificare il media

## @rules: media

```
<style type="text/css">  
@media screen {  
  h1 {color: black;}  
  p {color: red;}  
}  
@media print {  
  h1 {color: red;}  
  p {color: black;}  
}  
</style>
```

# Il media PRINT

Esistono particolari proprietà pensate per i “paged media”, come le stampanti:

```
@page {size: 210mm 297mm; margin: 30mm;}
```

```
table {page-break-inside: avoid;}
```

```
div.sommario {page-break-after: always;}
```

# @rules: import

Permette di includere un file CSS all'interno di un altro.

Favorisce un approccio *modulare*

```
<style type="text/css">  
@import url(mio_stile.css);  
</style>
```

Deve essere la prima delle regole

**Errore!**

```
<style type="text/css">  
h1 { color: black; }  
@import url(foglio_di_stile.css);  
</style>
```

# @rules: import

Si possono usare più regole @import

```
<style type="text/css">  
@import url(reset.css);  
@import url(fonts.css);  
@import url(layout.css);  
</style>
```

# @rules: import + media

@import può aver specificato anche il media

```
<style type="text/css">  
@import url(print.css) print;  
@import url(default.css) all;  
@import url(layout.css) print, screen;  
</style>
```

# Stylesheet alternativi

```
<link rel="stylesheet" type="text/css" href="stile.css">  
  
<link rel="alternate stylesheet"  
      title="Stile alternativo"  
      href="stile_alternativo.css" type="text/css">
```

Lo "switch" è fattibile, quando possibile, tramite l'interfaccia utente del browser.

Altrimenti occorre JavaScript

# Buchi e scappatoie

Lista dei buchi di IE e possibili soluzioni

<http://www.positioniseverything.net/explorerer.html>

# Bachi e scappatoie

## The underscore hack

The underscore ("\_") **is allowed** in CSS identifiers by the CSS2.1 Specification

Browsers have to ignore unknown CSS properties

MSIE 5+ for Windows ignores the "\_" at the beginning of any CSS property name

```
#menu {  
    position: fixed;  
    _position: absolute;  
    ...  
}
```

# Bachi e scappatoie

## Universal Selector Hack

Tutti i seguenti selettori sono considerati da IE, come se non avessero il "\*" davanti

```
* html      /* basic form */
* * body    /* whitespace between asterisks is significant */
* html body /* higher specificity than either of above */
```

In pratica, IE li interpreta così:

```
html
* body
html body
```

Bachi e scappatoie

Gli sporchi trucchi  
(dirty hacks)  
non vanno usati!

# Bachi e scappatoie

## Conditional comments

[http://msdn.microsoft.com/workshop/author/dhtml/overview/ccomment\\_ovw.asp](http://msdn.microsoft.com/workshop/author/dhtml/overview/ccomment_ovw.asp)

```
<!--[if IE]>
<style>
#footer .search
{
    margin-top: -1.6em;
}
</style>
<![endif]-->
```

# Bachi e scappatoie

## Conditional comments

Meglio ancora...

```
<!--[if IE]>  
<link type="text/css" href="ie_fix.css" rel="stylesheet">  
<![endif]-->
```

# Cascade

Per sapere come un elemento deve apparire il browser deve calcolare:

1. Quale media? (print, screen, aural)
2. Origine e peso
3. Calcola la specificità per risolvere eventuali conflitti
4. Viene applicata la regola più vicina all'elemento in questione (inline, embedded, linked)

# Concludendo - 1

Si sviluppa secondo le regole

poi

Si adattano le regole ai bug

# Concludendo - 2

1. Non usare DIV/SPAN inutili (no divitis)
2. Usare nomi di classi e ID significativi
3. Quando possibile non usare elementi generici

# Risorse

**[http://del.icio.us/corso\\_awa/risorse](http://del.icio.us/corso_awa/risorse)**

<http://css-discuss.incutio.com/>

<http://meyerweb.com/eric/css/tests/css2/>

<http://www.positioniseverything.net/easyclearing.html>

<http://blog.html.it/layoutgala/>

<http://www.famfamfam.com/lab/icons/silk/>

[http://www.snook.ca/technical/colour\\_contrast/colour.html](http://www.snook.ca/technical/colour_contrast/colour.html)

<http://typetester.maratz.com/>

<http://developer.yahoo.com/yui/grids/>