

CASCADING STYLE SHEET - introduzione

OutLine: struttura "semantica" del documento, data mediante (x)html markup (*elementi strutturali*)

Lay^ouT:

dato da regole di presentazione degli elementi, definite mediante specifiche CSS

CSS = ... css2 ... css2.1 ... css3 =

possibilità/necessità di determinare (+/- univocamente) la presentazione;

gli elementi presentazionali in html sono "deprecati" ()

permette migliore consistenza e manutenibilità del sito

estensione vasta, (effetti visivi, responsiveness, accessibilità)

specifica delle regole: **inline** ... **internal** ... **external**

CASCADE = le regole di un elemento sono "ereditate" da quelli ad esso interni; a meno di ricoprimenti delle (parti di) regole, specificate "piu` vicino" all'elemento interno (cercheremo di capire cosa vuol dire "piu` vicino").

CASCADING STYLE SHEET - introduzione

OutLine:

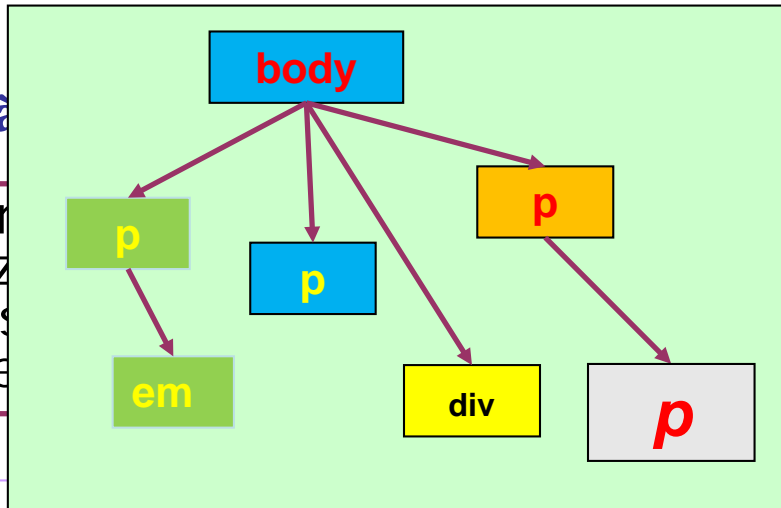
struttura "semantica" del documento, data mediante (x)html markup (*elementi strutturali*)

Lay^ouT:

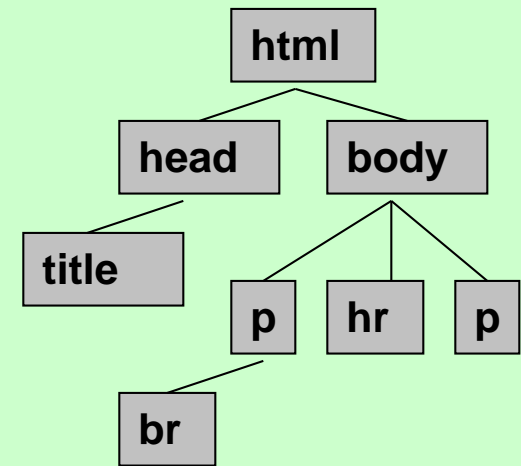
dato da regole di presentazione degli elementi, definite mediante spec. CSS

possibilità

gli elementi di presentazione
html sono "depre-



albero del documento



specifiche delle regole: **inline** ... **internal** ... **external**

CASCADE = le regole di un elemento sono "ereditate" da quelli ad esso interni; a meno di ricoprimenti delle (parti di) regole, specificate "piu' vicino" all'elemento interno ("che vuol dire piu' vicino?")

CASCADING STYLE SHEET

La specifica CSS definisce i modi con cui scrivere *regole di stile* ed applicarle ad una pagina, determinandone così la visualizzazione.

Una **regola di stile** è un insieme di definizioni (**comandi di stile**) che, assegnata ad uno o più elementi, indica al browser come dovrà avvenire la presentazione complessiva di quegli elementi.

Un comando di stile influenza **proprietà** (relative alla presentazione) degli elementi, quali la Font e lo stile del testo, i colori, il posizionamento, modificandone la visualizzazione di default propria del browser.

Un comando di stile ha la forma

proprietà: valore

REGOLA INLINE: definite direttamente nell'elemento che governa (attributo "style").

```
<body style="background-color: yellow; color: blue">
```

```
<h1 style="text-align: center; color: purple;
font-family: Arial, sans serif">stili inline
(h1 non standard)</h1>
```

```
<p style="color: black; font-size: 120%; text-align:
right">
```

The End.

```
</p>
```

Il documento sarà visualizzato con testo blue su fondo giallo

Header centrato, testo viola di font arial (o, in mancanza, una qualunque sans-serif disponibile)

Questo elemento p modifica il colore testo da blue (ereditato dall'elemento di cui è figlio: body) a nero; il testo sarà più grande del valore normale (ereditato) e sarà allineato a destra

**Comandi
separati da ;**

inlineStylesheet.html

```
<body style="background-color: yellow; color: blue">
<h1>stili inline</h1>anche h1 e` block-level
<h1 style="text-align: center; color: purple; font-
family: Arial, sans serif">stili inline ... </h1>
```

```
<table border="1" cellspacing="3" cellpadding="5">
<thead style="background-color: green">
```

```
<tr><td></td>
<th>articolo</th>
<th>prezzo</th> </tr> </thead>
```

```
<tbody>
<tr><td rowspan="2" style="vertical-align: top">
verdura</td>
```

Il blue/yellow del body resta

```
<td>zucchine</td>
<td>&euro;2,50</td> </tr>
```

```
<tr>
<td>asparagi</td>
<td>&euro;3,05</td> </tr>
```

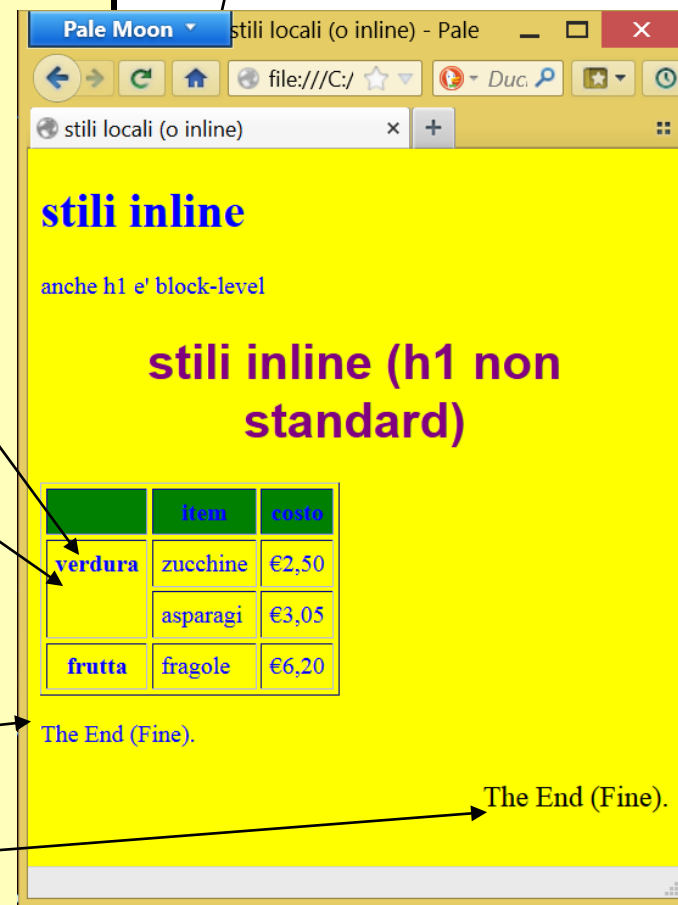
```
<tr>
<th>frutta</th>
<td>fragole</td>
<td>&euro;6,20</td>
</tr> </tbody> </table>
```

Un p normale e uno
che modifica alcune
proprietà

```
<p>The End...</p>
<p style="color: black; font-size: 120%; text-align:
right">The End...</p>
```

h1 normale (eredita le
impostazioni di body)

! testo fuori markup



A) internal stylesheet: insieme di regole definite nella `<head>` del documento, con il tag `<style>`

B) external stylesheet: insieme di regole definite in file a parte, che poi viene riferito con un elemento `<link>` del documento;

Per definire una regola si associa

- **un selettore** (elemento o “tipo di elemento” cui applicare la regola)
- **a un insieme di comandi di stile** racchiuso tra `{...}` (la regola:).

`selettore {property1: value1; ...; propertyn: valuen}`

Il documento che usa **queste regole** ha

- `<body>` formattato come indicato qui
- **ogni** elemento `<h1>` formattato come
- **ogni** elemento `<p>` formattato come

... se **non** vogliamo assegnare un certo stile a **tutte** le occorrenze di un certo elemento, dobbiamo usare selettori più particolareggiati.

```
body {background: yellow;
      color: blue;
}

h1 {text-align: center;
    color: purple;
    font-family: Arial,sans serif;
}

p {color:black;
   font-size: 120%;
   text-align: right;
}
```

`external.css`

con	le regole {...} si applicano a
<code>h1 {...}</code>	tutte le occorrenze dell'elemento h1
<code>h1, h2 {...}</code>	tutte le occorrenze di h1 e tutte quelle di h2
<code>h1 em {...}</code>	tutte le occorrenze dell'elemento em dentro ad un h1
<code>h1.special {...}</code>	<u>solo</u> ad una classe di elementi h1: <u>cioe` alle occorrenze dell'elemento</u> <code><h1 class="special">...</h1></code>
<code>.centrato {...}</code>	le occorrenze di qualsiasi elemento con <code>class="centrato"</code> esempio: <code><p class="centrato">...</p></code>
<code>th[rowspan] {...}</code>	elemento th in cui l'attributo rowspan e` assegnato
<code>th[rowspan="2"] {...}</code>	elemento th in cui l'attributo rowspan e` assegnato a 2
<code>sel[attr~="value"] {...}</code>	in generale, regola gli elementi sel aventi attributo <code>attr</code> contenente la parola value
<code>p[color\$="e"] {...}</code>	... p in cui l'attributo color e` assegnato a un colore che finisce per e
<code>p[color^="m"] {...}</code>	... p in cui l'attributo color e` assegnato a un colore che inizia per m
<code>p[color*="rt"] {...}</code>	... p in cui l'attributo color e` assegnato a un colore che contiene rt
<code>XX#crossed {...}</code>	elemento XX con <code>id="crossed"</code> (NB. unicit` di id)
<code>h1 > em {...}</code>	occorrenze di un elemento em in un h1 , come figlio
<code>div + h1 {...}</code>	occorrenze di h1 subito dopo un div (primo fratello)
<code>em:lang(langSpec)</code>	occorrenze di em in un documento basato su langSpec

pseudo-classi

Una pseudo-classe e` una specifica aggiuntiva, valida per certi elementi. Viene usata per regolare l'occorrenza

- di uno stato particolare dell'elemento

- l'elemento e` in focus, per effetto di una selezione attraverso la tastiera, o un'interazione con l'utente – campo di una form selezionato ...,
- il link viene "hovered", o non e` stato ancora usato, o e` stato gia` visitato ...

- o di una relazione tra l'elemento e il documento

- l'elemento XX e' il primo discendente dell'elemento in cui risiede, o e` l'n-simo, o l'n-ultimo

XX:focus {color ... background_image ...}

a:link, a:visited, a:hover, tr:hover,
 XX:focus, a:active, XX:first-child, XX:nth-last-child()

pseudo-elementi

Uno pseudo-elemento e` una specifica aggiuntiva, relativa a parti del contenuto di un elemento: prima linea, prima lettera, parte selezionata, area prima o dopo del contenuto ...

XX:before, XX:after,

XX:first-line, XX:first-letter, XX:spelling-error

externalStylesheet.html

```
<?xml version = "1.0"?>
... <html ...
<head><title>stili esterni</title>
<link rel="stylesheet" href="external.css"
type="text/css" /></head>

<body>
<h1>...external</h1>anche h1 e` block-level
<h1 class="speciale">(h1 non standard)</h1>

<table border="1" cellspacing="3" cellpadding="5">

<thead>
  <tr>...</tr>
</thead>

<tbody>
  <tr>
    <th rowspan="2">verdura</th>
    <td>zucchine</td>
    <td>&euro;2,50</td>
  </tr>
  ...
</tbody>
</table>

<p>The End.</p>
<p class="spostato">The End.</p>
</body></html>
```

```
body {background-color: yellow;
      color: blue}
/* COMMENTO */
thead {background-color: green;}

h1.speciale {
  text-align: center;
  color: purple;
  font-family: Arial,sans serif;
}
/* COMMENTO */
p.spostato {color:black;
            font-size: 120%;
            text-align: right;}

th[rowspan="2"] {
  vertical-align: top; }
```

external.css



internalStylesheet.html

```
<?xml version = "1.0"?>
  ... <html ...
<head><title>stili esterni</title>
<style>
body {background-color: yellow; color: black;}

thead {background-color: green;}

h1.speciale {
  text-align: center;
  color: purple;
  font-family: Arial,sans serif;
}
p.spostato {
  color:black;
  font-size: 120%; text-align: right;}

</style>
</head>

<body>...
</body></html>
```

Esperienza per vedere se
"frutta e' allineato in alto.
(es. aggiungi contenuto dove
c'e' fragole - la cella si
ingrandisce)



```
...      .inTesta {vertical-align: top;}

.inFondo {vertical-align: bottom;}

td.destra {text-align: right;}

td#soloQuesto { text-align: center;}

*#unico {text-align: right; color:black;}

...
<table ...>
<thead>...</thead>    <tbody>
  <tr>
    <th class="inTesta" rowspan="2">verdura</th>
    <td>zucchine</td><td>&euro;2,50</td>    </tr>
  <tr>
    <td>asparagi</td>
    <td class="destra">&euro;3,05</td>    </tr>
  <tr>
    <th class="inFondo">frutta</th>
    <td id="soloQuesto">fragole<br />buonine</td>
    <td id="unico">&euro;6,20</td>
  </tr>
</tbody></table> ...
```



stili interni

anche h1 e' block-level

stili internal (h1 non standard)

	articolo in questione	e suo prezzo
verdura	zucchine	€2,50
	asparagi	€3,05
frutta	fragole buonine	€6,20

The End.

The End.

```
...      .inTesta {vertical-align: top;}

.inFondo {vertical-align: bottom;}

td.destra {text-align: right;}

td#soloQuesto { text-align: center;}

*#unico {text-align: right; color:black;}

...
<table ...>
<thead>...</thead>    <tbody>
  <tr>
    <th class="inTesta" rowspan="2">
      <td>zucchine</td><td>&euro;2,50</td>
  <tr>
    <td>asparagi</td>
    <td class="destra">&euro;3,05</td>
  <tr>
    <th class="inFondo">frutta</th>
    <td id="soloQuesto">fragole<br />
    <td id="unico">&euro;6,20</td>
  </tr>
</tbody></table>
```

VERIFICA

internalStyleSheet2.html

contiene errori rilevati dal validatore

internalStylesheet2.check.html
e' corretto (... forse)

***#unico {text-align: right; color: black}**
stabilisce una regola che vale per l'elemento che ha
attributo id="unico"

Come sempre

- **l'occorrenza** dell'elemento con attributo id=XXX deve essere **unica** nel documento;
- un elemento cui si applica una regola, prima di tutto *eredita* lo stile definito per l'elemento di cui e' figlio: la regola modifica solo quelle proprietà che sono in essa esplicitamente riferite (in questo caso colore e allineamento del testo)

... milioni di stylesheet properties

Box model - Microsoft Internet Explorer

Back Forward Stop Reload Favorites Media File Edit View Favorites Tools Help

Address file:///C:/Documents%20and%20Settings/m/Desktop/css/css2/box.html#value-def-border-width

type, which make take one of the following.

none
No border. This value forces the computed value of `'border-width'` to be '0'.

hidden
Same as 'none', except in terms of [border conflict resolution](#) for [table elements](#).

dotted
The border is a series of dots.

dashed
The border is a series of short line segments.

solid
The border is a single line segment.

double
The border is two solid lines. The sum of the two lines and the space between them equals the value of `'border-width'`.

groove
The border looks as though it were carved into the canvas.

ridge
The opposite of 'groove': the border looks as though it were coming out of the canvas.

inset
The border makes the entire box look as though it were embedded in the canvas.

outset
The opposite of 'inset': the border makes the entire box look as though it were coming out of the canvas.

All borders are drawn on top of the box's background. The color of borders drawn for values of 'groove', 'ridge', 'inset', and 'outset' depends on the element's `'color'` property.

[Conforming HTML user agents](#) may interpret 'dotted', 'dashed', 'double', 'groove', 'ridge', 'inset', and 'outset' to be 'solid'.

'border-top-style', 'border-right-style', 'border-bottom-style', 'border-left-style'
Value: `<border-style>` | [inherit](#)
Initial: none
Applies to: all elements
Inherited: no
Percentages: N/A
Media: [visual](#)

'border-style'
Value: `<border-style>{1,4}` | [inherit](#)
Initial: [see individual properties](#)

La specifica CSS dice quali proprietà sono applicabili a quali elementi; il CSS validator verifica la sintassi delle regole del documento (similmente alla *markup validation*)

VEDI [directory XHTML.CSS.2/VALIDAZIONE](#)

qualche indirizzo (tra I molti)

<https://www.w3.org/TR/css-box-3/>
<https://www.w3.org/TR/css-backgrounds-3/#border-width>
<https://www.tutorialrepublic.com/css-reference/css3-properties.php>

... miliardi di stylesheet properties

- font-family (es. con un nome, come Times New Roman, Arial, ... o generiche come serif, sans serif, monotype, cursive, fantasy)
- font-size (dimensione font)
- font-style (valori definiti, es. normal, oblique, italics)

- color (text color: specifica di un colore)
- background-color
- background-image
- background-attachment (scroll o fixed immagine di background)

- text-align (valori definiti – es. *left*, ...)
- text-transform (es: lowercase, uppercase)
- list-style-type (es. disk, square, lower-roman)
- letter-spacing, word-spacing

- width, height, margin, padding, border (misure - dimensione - layout)

- top (y coord. della cima dell'elemento)
- left (x coord. della cima dell'elemento)
- z-index (precedenza: se sovrapposti, quali elementi appaiono sopra e quali sotto?)
- float (allineamento rispetto ad altri elementi – es. testo intorno ad un elemento)

dimensioni

Ci sono misure cosiddette assolute e cosiddette relative, basate su valori numerici o enumerati...
Le definizioni di dimensione riguardano la grandezza dei caratteri, o caratteristiche di resa grafica di elementi (es. la larghezza di un elemento block-level).

Assolute rispetto a che cosa? Relative a che cosa?

- px pixel (evidentemente dipende dallo schermo ...)
- em font size della **font in uso nell'elemento padre** (*relevant*)
- ex altezza della x nella relevant font
- % percentuale della dimensione associata all'elemento padre
- in inches
- mm
- cm
- pt point (*1/72th of an inch*)
- pica 12 punti (tradizionale nell'editoria)
- xx-small | x-small | small | medium | large | x-large | xx-large
valori predefiniti, sensibili alle impostazioni dell'applicazione utente
- larger | smaller
(dipendono dalla dimensione del carattere nell'elemento padre)

Specifica colori

specifica base rgb

#**XXXXXX**

6 cifre esadecimali (2 per ciascun colore base)

16 milioni di colori

```
p {color: #000000} /* nero */           p {color: #FF0000}
p {color: #A01024} /* ? */
p {color: #f00} /* notazione veloce per ff0000 */
p {color: rgb(0,0,255)} /* 0-255 */
p {color: rgb(50%,10%,33%)} /* 0.0-100.0 % */
```

non e` garantita una resa omogenea di ciascun colore su qualsiasi macchina

palette web safe

Inventata per diminuire i problemi derivanti dall'uso del dithering nella visualizzazione di immagini su monitor con profondita` di colore "scarsa" (dithering? profondita` eh?)

#**XXXXXX**

- 6 cifre esadecimali

- ciascuna coppia e` uno tra 00, 33, 66, 99, CC, FF: 216 colori

- trattamento dei colori cosi` specificati *dovrebbe* esser + omogeneo

"named colors" *Insieme iniziale di 16 ... poi esteso ...*

aqua, black, blue, fuchsia, gray, green, lime, maroon,
navy, olive, purple, red, silver, teal, white, and yellow

```
p {color: maroon} /* #800000 */
```

(<https://htmlcolorcodes.com/color-names/>)

(se volete dare nome ad un colore ...<https://colornames.org/>)

Modello di Box (1/2)

Ogni elemento e` racchiuso in una box, in cui

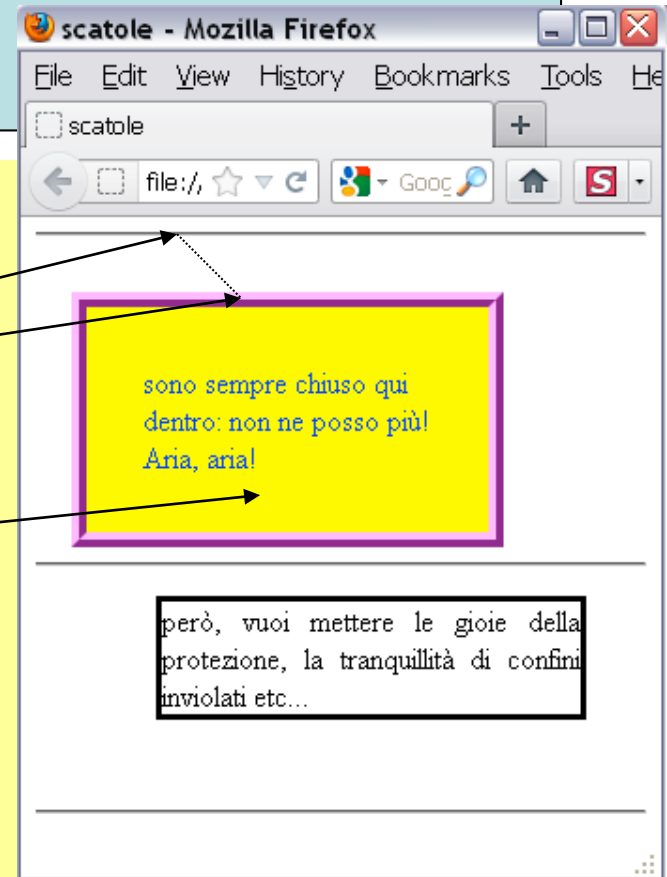
border = parte potenzialmente visibile, intorno al contenuto dell'elem.

padding = area tra il contenuto e il bordo

margin = area tra l'elem. e gli altri elem. vicini

...

```
.myBox {background: yellow;color: blue;
border-width: 8px;
border-style: ridge;
border-color: fuchsia;
margin-top: 30px;
margin-left: 5mm;
margin-right: 20mm;
padding: 30px;
}
</style></head>
<body>
<hr />
<div class="myBox">
sono ...</div>
<hr />
<div style="margin: 1ex 2em 3ex 4em;
border:solid;text-align:justify;">
...</div>
<hr />
</body></html>
```



margin-top -right -bottom -left

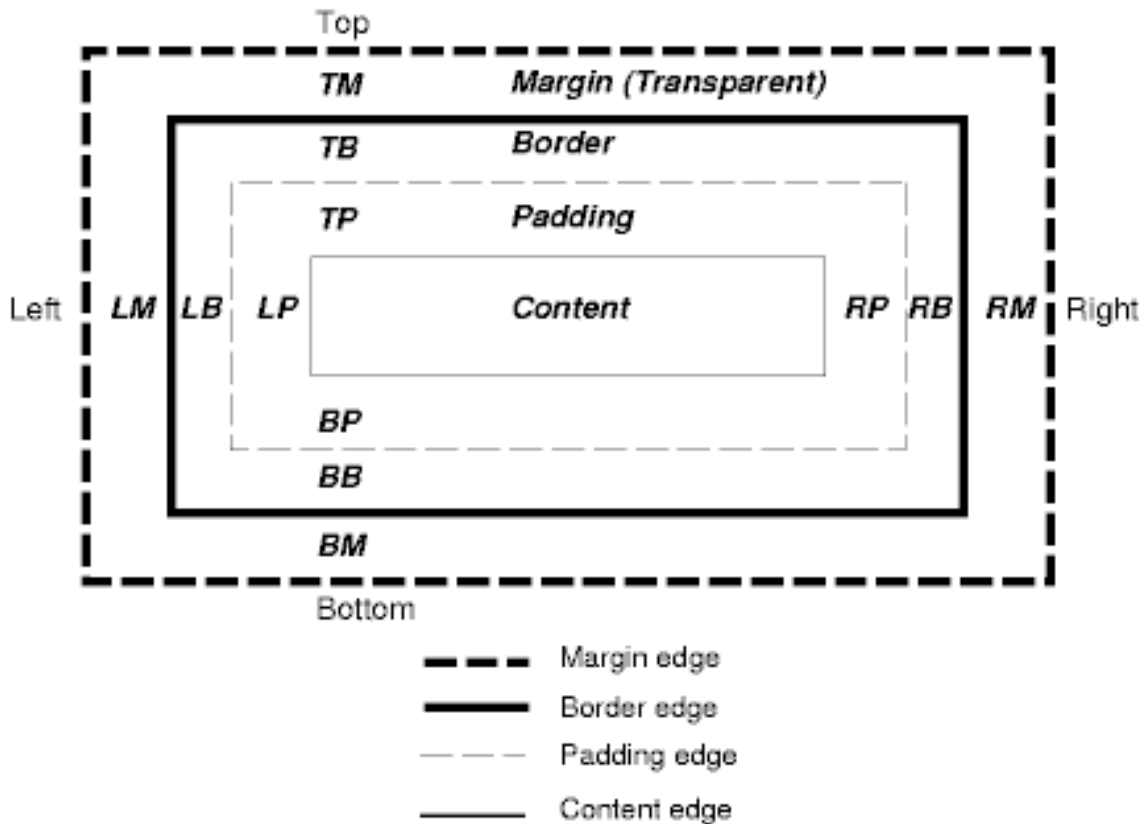
box.html

Ogni elemento e' racchiuso in una box, in cui

border = parte potenzialmente visibile, intorno al contenuto dell'elem.

padding = area tra il contenuto e il bordo

margin = area tra l'elem. e gli altri elem. vicini



Specifica CSS box

Legenda

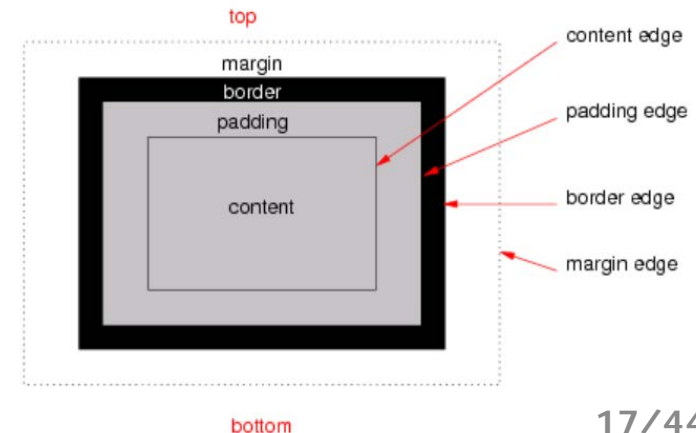
TM/TB/PP = Top

Margin/Border/Padding

B=Bottom

L=Left

R=Right



CSS perche'?

Vantaggi

- Chiara **separazione** tra contenuto, con la sua strutturazione, e regole di presentazione,
 - delineate (proposte/decise) prima dall'autore che dal fruitore e/o browser
- Regole di presentazione
 - definibili **rigorosamente**
 - **condivisibili** da piu` documenti (si uniforma il *look-and-feel* del sito web e lo si rende piu` **facilmente mantenibile e modificabile**)
 - permettono **piu` compatibilita`** (migliore gestibilita`) con le applicazioni web che eseguono la presentazione
 - **visualizzazione**,
 - **indipendenza** dal dispositivo (sviluppando regole distinte ... device palmare, telefono, tablet, tv)
 - **tecnologie assistive** (sintesi vocale, braille)

Svantaggi

- **Complessita`** (per autori e tools ...)

Posizionamento di elementi

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>primo uso di position</title>
</head>
<body>
```

```
<hr />
```

```
<p>
```

```
In questo testo c'e` una
<span style="position: relative;
        top: 10ex; left: 1em">
```

```
caduta di stile
```

```
</span>
```

```
(ah ah ah ...)
```

```
</p>
```

```
<hr />
```

```
<div style="color:blue">
```

```
Questo testo continua da dove doveva continuare ma
anche per lui
```

```
<span style="position: absolute; top: 10ex; left: 1em">
<em>cade</em></span>
```

```
qualcosa.</div>
```

```
<hr />
```

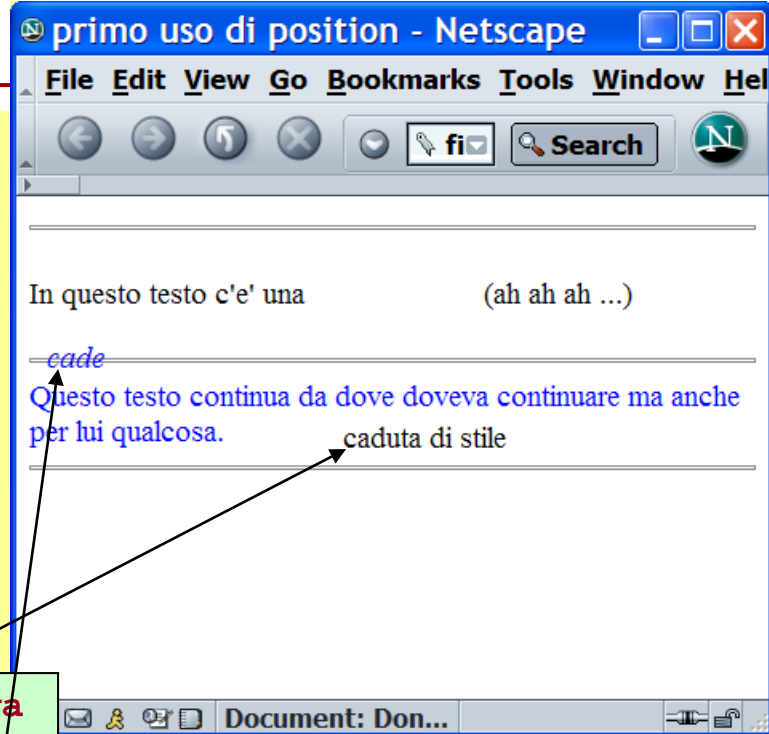
```
</body>
```

```
</html>
```

Posizione della caduta, **relativa**
= calcolata rispetto a dove
naturalmente sarebbe

Posizione della caduta,
assoluta=relativa a
(0,0)

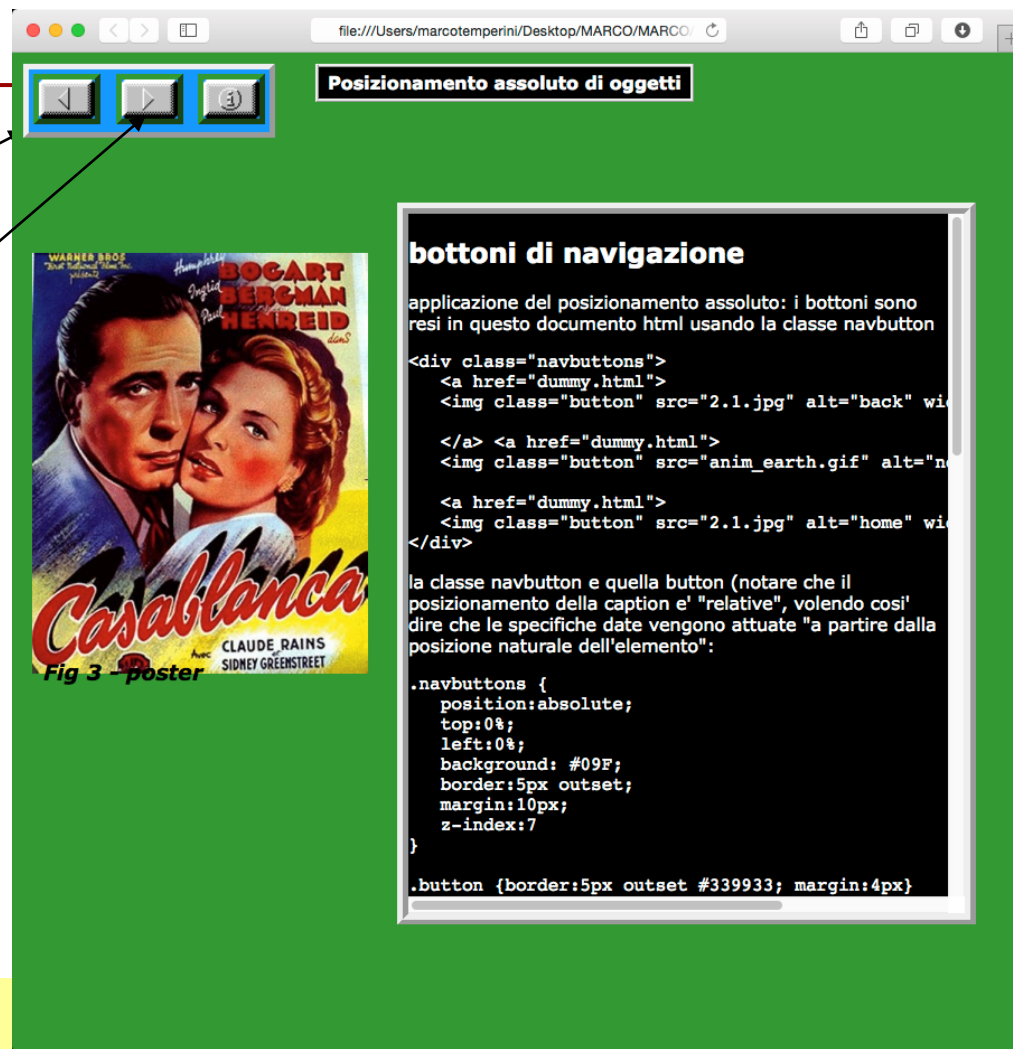
firstPositioning.html



positioning.html (1/4)

positioning.css

```
.navbuttons {  
  position:absolute;  
  top:0%;  
  left:0%;  
  background: #09F;  
  border:5px outset;  
  margin:10px;  
  z-index:7  
}  
  
.button {  
  border:5px outset #339933;  
  margin:4px  
}
```

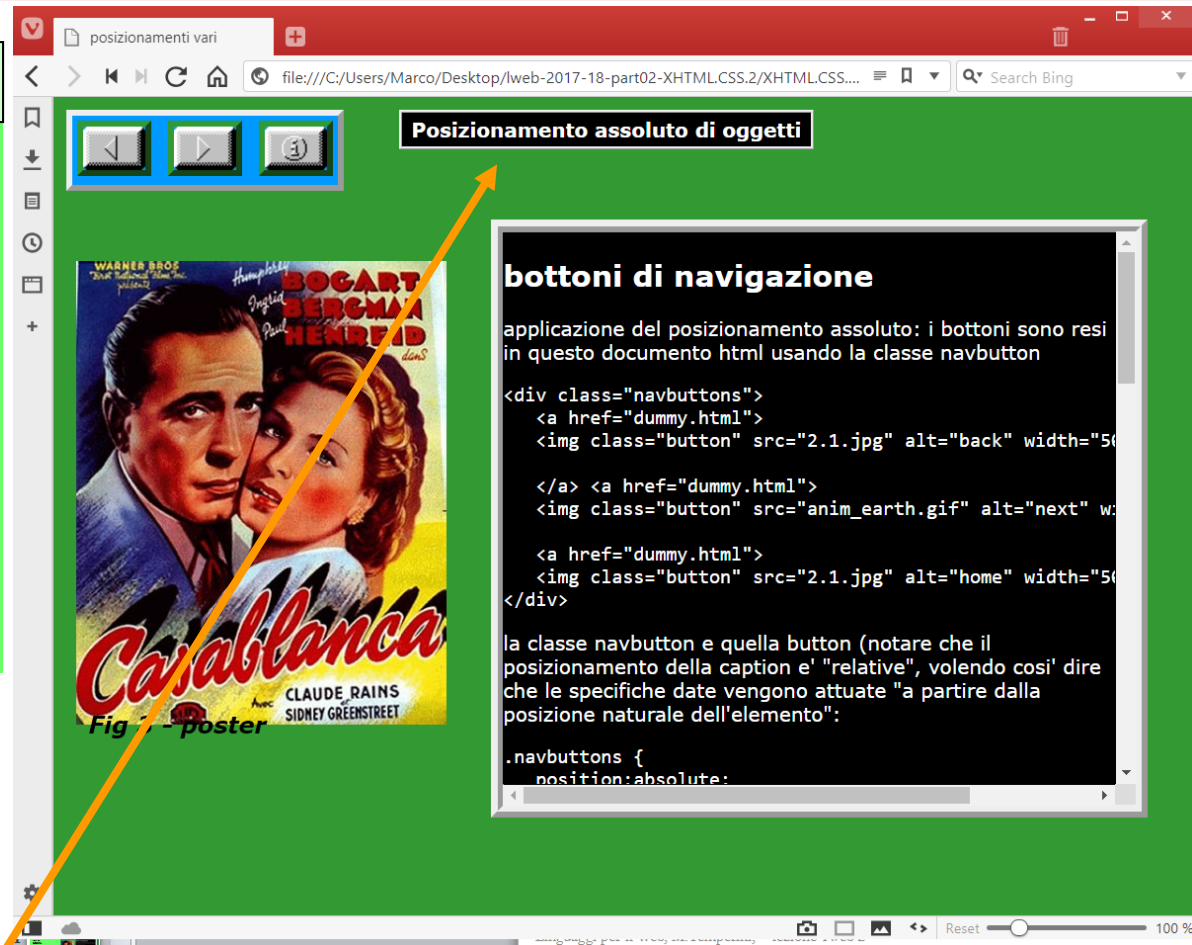


```
<div class="navbuttons">  
  <a href="dummy.html">  
    </a>  
    <a href="dummy.html">  
      </a>  
      <a href="dummy.html">  
        </a>  
    </div>
```

positioning.css

```
.headerText {  
  position: absolute;  
  top: 0%;  
  left: 30%;  
  background: black;  
  border-style: ridge;  
  margin-top: 10px;  
  padding: .5ex .5em;  
  text-align: center;  
  font-weight: bold;  
  z-index: 1  
}
```

```
<div class="headerText">  
  Posizionamento assoluto di oggetti  
</div>
```



positioning.css

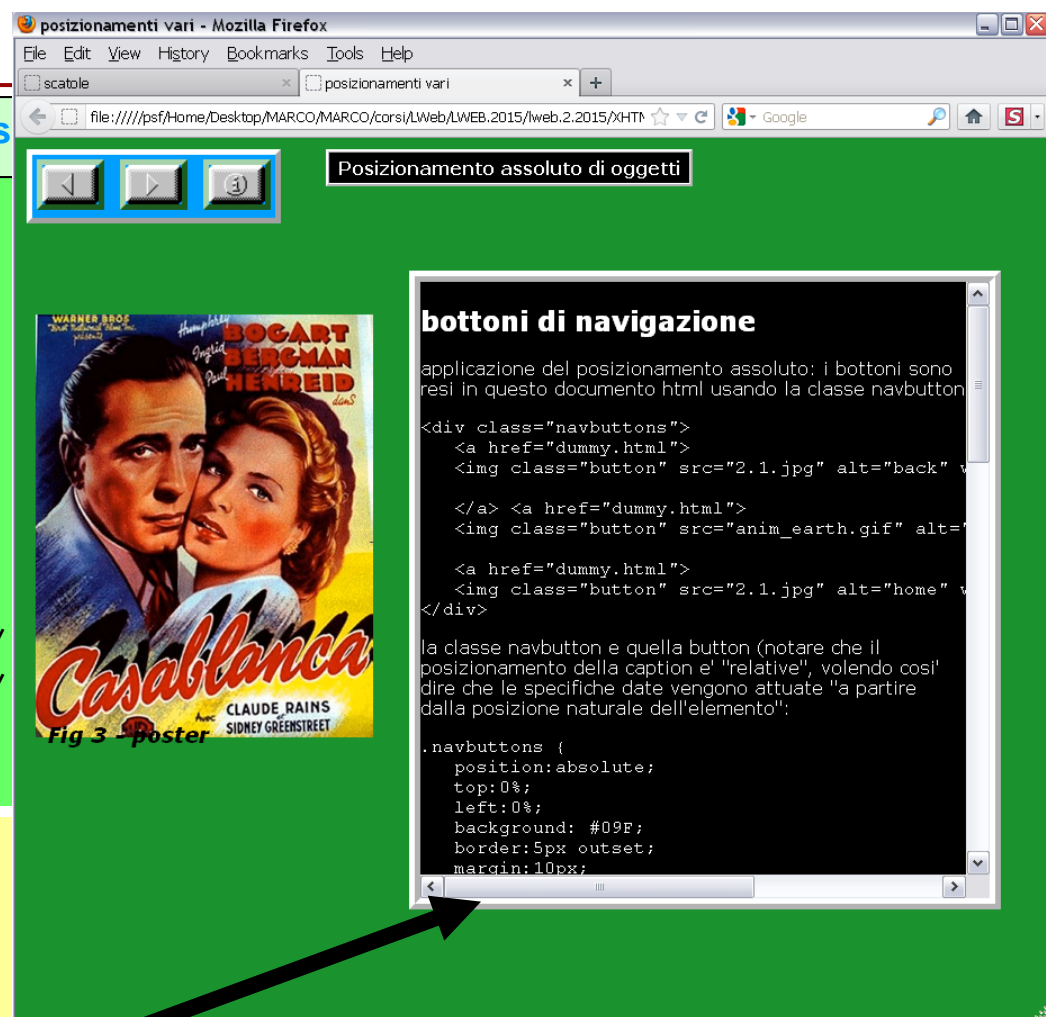
```
.picture {
    position:absolute;
    top:20%;
    left:2%;
    z-index:2
}
.caption {
    position:relative;
    top:-1.5ex;
    left:0.5em;
    font-style:italic;
    font-size: 120%;
    font-weight: bold;
    color: black;
    z-index:3
}
```

```
<div class="picture">
<br />
<div class="caption">Fig 3 - poster</div>
<!-- per provare qual e` la posizione naturale
<div>Fig 3 - poster</div>
-->
</div>
```



```
.text {
  position: absolute;
  top:15%;
  left:38%;
  width:55%;
  height:70%;
  border:10px ridge;
  padding:0%;
  overflow:auto;
  background: black;
  /*background-image:url(img.jpg); */
  /*background-attachment:fixed; */
  z-index:4;
}
```

```
<div class="text">
<h2>bottoni di navigazione</h2>
<p>
applicazione ...</p>
<pre>
<div class="navbuttons">
  <a href="dummy.html">
    
    ...
  </a>
</div>
...</div>
```



import

Comandi di stile possono essere inclusi in un file .css, reperendoli (importandoli) da un altro)

```
@import "mystyle.css";
```

```
@import url("mystyle.css");
```

uso di import in file css...

mystyle.css

```
@import url("immagini.css");  
@import url("classi.css");  
body {  
    background-color: yellow;  
    color: blue  
}  
...
```

mydocument.html

```
<html xmlns="http://...  
  <head>  
    <title>styles...</title>  
    <link rel="stylesheet" href="mystyle.css"  
    type="text/css" ... />  
  </head>
```

mydocument.html

```
<html xmlns="http://...  
  <head>  
    <title>styles...</title>  
    <link rel="stylesheet" href="mystyle.css" type="text/css" ... />  
    <link rel="stylesheet" href="immagini.css" type="text/css" ... />  
    <link rel="stylesheet" href="classi.css" type="text/css" ... />  
  ...</head>
```

non usando import, si possono inserire link ai vari file css direttamente nel documento MA la sequenza dei comandi di stile allora sarà diversa da quella ottenuta in mystyle.css con le import (comandi ridondanti e l'uno ricoprente l'altro appariranno in ordine diverso e facendo ottenere un risultato diverso da quello precedente)

Regole di stile in conflitto vengono “pacificate” applicando un algoritmo di calcolo della precedenza. E si parla di “specificità” delle regole applicate, per decidere quali applicare in caso di conflitti.

- Principio base: **Author > User > User Agent**

L'autore specifica delle regole; queste hanno la precedenza su quelle eventualmente definite dallo user (personal style sheet, definita dallo *user*, cioè da “chi sta usando il browser ...”). Le regole dello user hanno poi precedenza su quelle definite nello user agent (default, in mancanza di scelte user-defined).

- Se uno dei tre ambienti ha definito ... { ... **!important** } la regola, ha la precedenza.

- A parità di precedenza, in un medesimo ambiente vince l'ultima regola specificata.

- **UL LI > LI** selettore più **specifico** => maggiore precedenza

<https://blog.hubspot.com/website/css-specificity>,
<https://web.dev/learn/css/specificity>
e w3schools

Comandi di stile possono determinare la presentazione del documento in base al tipo di “media” (aural, tactile, visual) in azione

possibili media: **all**, **speech** (speech synthesizers), **braille**, **embossed** (braille print), **handheld**, **print**, **projection**, **screen**, **tty** (caratteri fissi, terminals), **tv**.

Esempio - 0 -

```
@media print {
    body { font-size: 10pt }
}
@media screen {
    body { font-size: 12pt }
}
```

In CSS3 anche media query - 3 -

```
monitor
media = screen
      and (min-width: 800px)
      and (max-width: 1280px)

telefonino
media = screen
      and (max-device-width: 480px)
```

esempio di applicazione di una media query

```
<link rel=stylesheet" type="text/css"
media="screen and
      (max-device-width: 480px)"
href="telefonino.css" />
```

Inserimento come stile interno - 1 -

```
<style type="text/css">
  media print {
    body{ font-family:
           Times New Roman, serif;
          font-size: 12pt; color: #000;
          background-color: white;
          !important;
        }

    .header {display: none}

    .footer {display: none}

    li {font-family:
        Courier New, sans serif;
       }
... continua <style> ...
```

Inserimento come stile esterno - 2 -

```
<head>
  <title>Link to a target medium</title>
  <link rel="stylesheet" type="text/css"
        media="print, handheld" href="foo.css" />
</head>
```

CSS3 miscellanea

La specifica CSS3 e` in continua evoluzione (<http://www.w3.org/Style/CSS/current-work>); in pratica uno standard, pronto per essere utilizzato stabilmente nei nuovi browser.

Un quadro riassuntivo (la situazione dei vari moduli, da “raccomandazioni” giu` fino a “working draft”) e` in http://www.w3.org/standards/techs/css#w3c_all

Esempio: per scaricare font e usarle

```
@font-face {  
    font-family: generica;  
    src: url(http://www.lottafonts.com/fonts/generic.ttf)  
}  
  
...  
p {font-family: generica}
```

Esempio: pseudo-classi strutturali

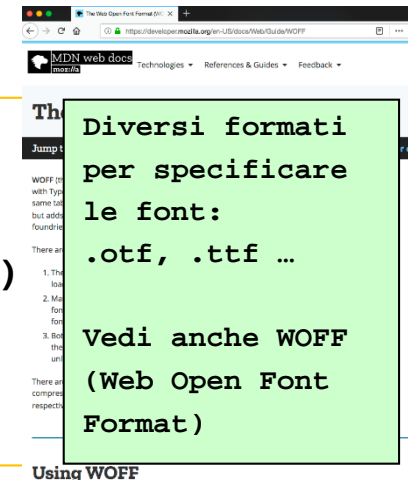
```
p:last-child {  
    color: blue;  
    text-decoration: underline;  
    border-bottom: solid thin maroon;  
} /* ultimo figlio del proprio genitore */  
  
p:nth-child(n) ...  
/* n-esimo figlio del proprio gen. */
```

Esempio: stato di selezione per elementi di input (vedi FORM piu` tardi nel corso)

```
input:checked {  
    color: maroon;  
    background-color:  
        yellow;  
}
```

(un'occhiata qui?

<https://developer.mozilla.org/en-US/docs/Web/CSS/:checked>)



Preferenze dell'autore e misure relative (1/6)

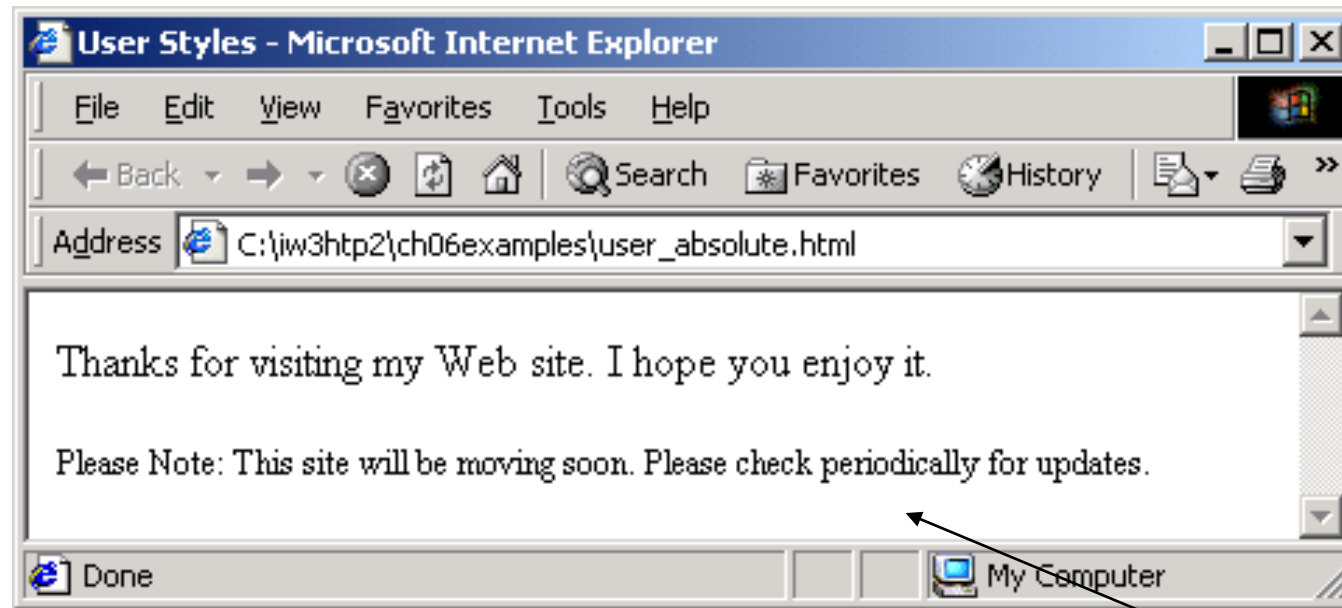
(NB - from Deitel et al, Internet & World Wide Web How to Program, www.deitel.com)

```
1  <?xml version="1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
...
9  <html xmlns="http://www.w3.org/1999/xhtml">
10     <head><title>User Styles</title>
...
12     <style type="text/css">
14         .note { font-size: 9pt }
16     </style>
17 </head>
18
19 <body>
20
21     <p>Thanks for visiting my Web site. I hope you enjoy it.
22     </p><p class="note">Please Note: This site will be
23     moving soon. Please checkperiodically forupdates.</p>
```

“Author defined style sheets”
Definite nella pagina, dall'autore

Questa specifica si impone su eventuali specifiche
definite dall'utente (chi usa il browser)

Preferenze dell'autore e misure relative (2/6)



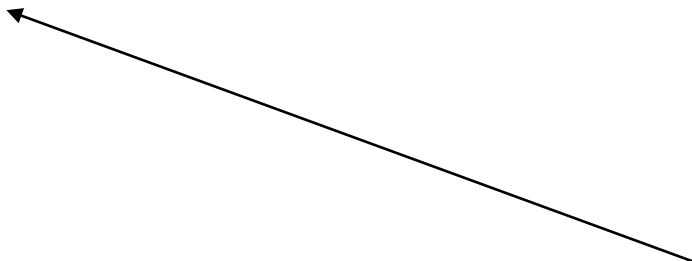
Secondo le specifiche date dall'autore:
carattere piu` piccolo, indicato in misura
assoluta

Author > User > User Agent

(NB - from Deitel et al, Internet & World Wide Web How to Program, www.deitel.com)

Preferenze dell'autore e misure relative (3/6)

```
1  /* Fig. 6.17: userstyles.css */
2  /* A user stylesheet          */
3
4  body    { font-size: 20pt;
5           color: yellow;
6           background-color: #000080 }
```

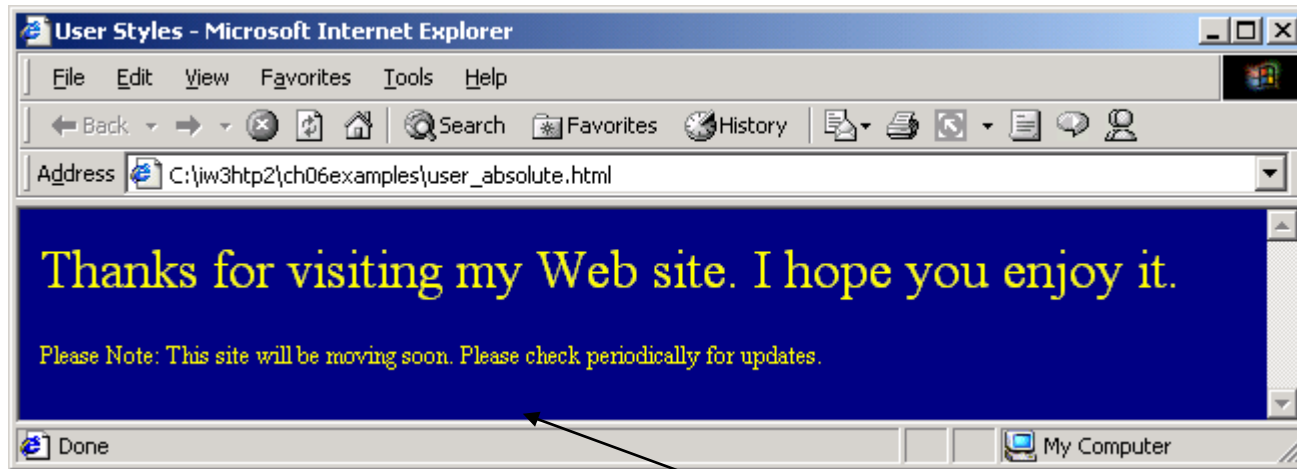


Esempio di stile definito dallo “user” – di solito stylesheet esterna, assegnata lavorando sulle opzioni del browser

(NB - from Deitel et al, Internet & World Wide Web How to Program, www.deitel.com)

Preferenze dell'autore e misure relative (4/6)

user stylesheets



L'utente ha detto che preferisce caratteri grandi; la prima riga non ha specifiche particolari date dall'autore: viene resa con stili di default del browser e con gli stili dettati dallo user;

La seconda riga viene resa usando gli stili dell'autore, perché l'autore li ha specificati e perché se ci sono hanno la precedenza ...

(NB - from Deitel et al, Internet & World Wide Web How to Program, www.deitel.com)

Preferenze dell'autore e misure relative (5/6)

```
1  <?xml version="1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
5  <!-- Fig. 6.20: user_relative.html - User styles      -->
7
8  <html xmlns="http://www.w3.org/1999/xhtml">
9      <head>
10         <title>User Styles</title>
11
12         <style type="text/css">
13
14             .note { font-size: .75em }
16         </style>
17     </head>
<body>
<p>Thanks for visiting my Web site. I hope you enjoy it.
</p><p class="note">Please Note: This site will be
moving soon. Please check periodically for updates.</p>
</body></html>
```

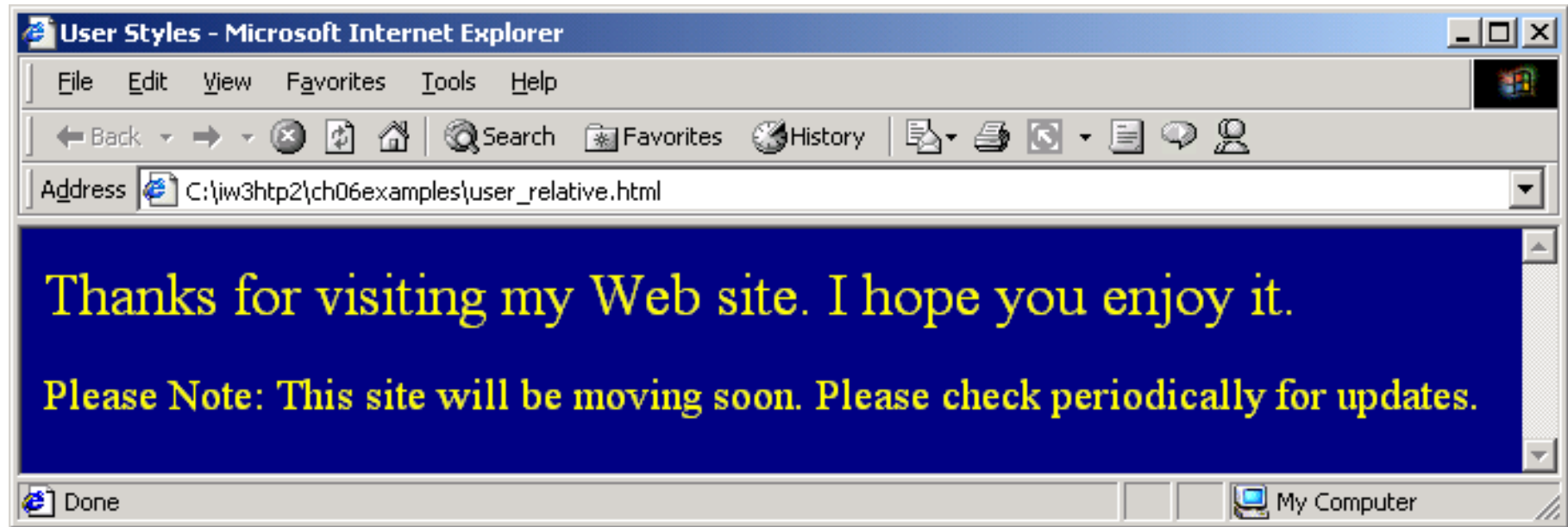
L'autore dice che una nota deve essere resa con carattere di dimensione al 75% del normale

Usare misure relative fa sì che la richiesta dell'autore (carattere più piccolo per le note) e quella dell'utente (in generale caratteri più grandi) si possano contemperare

Cioè è MEGLIO

Preferenze dell'autore e misure relative (6/6)

user stylesheets



.note ha precedenza sulla preferenza dell'utente; non ci sono altre specifiche da parte dell'autore, quindi il colore e la dimensione base dei caratteri rimangono come vuole lo user.

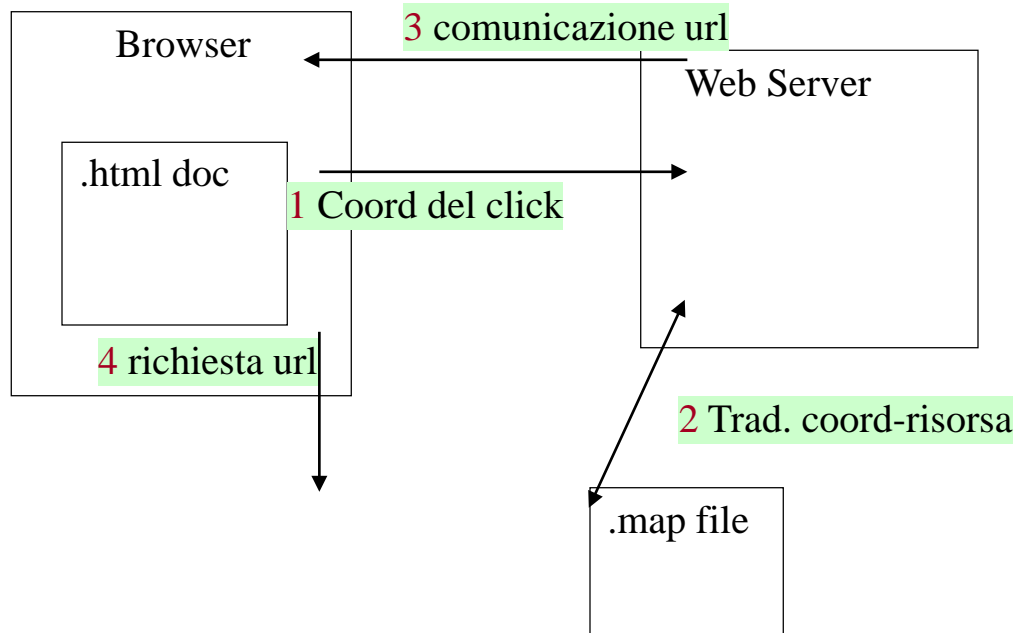
(NB - from Deitel et al, Internet & World Wide Web How to Program, www.deitel.com)

Immagini cliccabili (map)

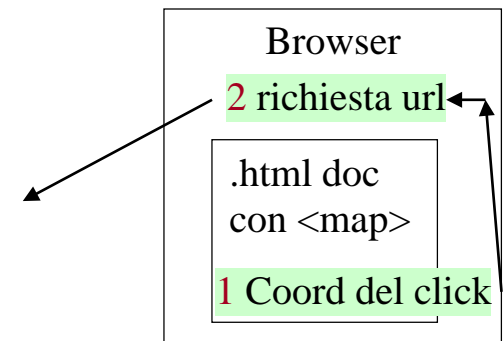
e' possibile definire **aree** di un'immagine come collegamenti.

Quando si fa click su un'immagine, il browser invia le coordinate del punto su cui si e' fatto click. Se queste coordinate sono nell'area prevista per un link, questo viene attivato.

La processazione dell'evento puo` essere server-side o client-side.



1 il browser consulta l'elemento map per scoprire quale url deve chiedere. E poi la chiede.



`clickableImage/clicking.img.html`
(client-side ...)

Appendice sui selettori CSS

enumerazione ed esempi di selettori, anche
per pseudo-classi e pseudo-elementi

Plain Selector

Seleziona tutti gli elementi di un determinato tipo.

```
p {  
  color: blue;  
}  
  
h1 {  
  font-size: 24px;  
}
```

Class Selector

Seleziona tutti gli elementi con una determinata classe.

```
.titolo {  
  font-weight: bold;  
}  
  
.avviso {  
  background-color: yellow;  
}
```

ID Selector

Seleziona un elemento specifico in base al fatto che l'elemento ha un certo attributo ID.

```
#intestazione {  
    text-align: center;  
}
```

```
#pulsante {  
    background-color: red;  
}
```

Universal Selector

Seleziona tutti gli elementi della pagina.

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

```
* {  
    font-family: Arial, sans-serif;  
}
```

Descendant Selector

Seleziona gli elementi discendenti di un altro elemento.

```
div p {  
    color: gray;  
}  
  
article span {  
    font-style: italic;  
}
```

Child Selector

Seleziona solo i figli diretti di un determinato elemento.

```
ul > li {  
    list-style-type: square;  
}  
  
nav > a {  
    text-decoration: none;  
}
```

Adjacent Sibling Selector

Seleziona il primo elemento fratello immediatamente successivo a un altro.

```
h1 + p {  
  font-size: 18px;  
}
```

```
img + span {  
  color: red;  
}
```

General Sibling Selector

Seleziona tutti gli elementi fratelli successivi di un determinato elemento.

```
h2 ~ p {  
  color: green;  
}
```

```
label ~ input {  
  border: 1px solid gray;  
}
```

Attribute Selector)**

Seleziona elementi in base a un attributo specifico.

```
input[type="text"] {  
    border: 2px solid blue;  
}
```

```
a[target="_blank"] {  
    color: orange;  
}
```


:hover

Attiva lo stile quando il mouse passa sopra l'elemento.

```
button:hover {  
    background-color: lightblue;  
}
```

:focus

Si applica quando un elemento riceve il focus (es. un campo di input selezionato).

```
input:focus {  
    border: 2px solid green;  
}
```

:active

Si applica quando un elemento viene cliccato.

```
``css  
a:active {  
    color: red;  
}
```

:visited

Si applica ai link già visitati dall'utente.

```
a:visited {  
    color: purple;  
}
```

`:first-child`

Seleziona il primo figlio di un elemento padre.

```
li:first-child {  
    font-weight: bold;  
}
```

`:last-child`

Seleziona l'ultimo figlio di un elemento padre.

```
p:last-child {  
    color: blue;  
}
```

`:nth-child(n)`

Seleziona l'ennesimo figlio di un elemento.

```
li:nth-child(3) {  
    color: red;  
}
```

`:nth-last-child(n)`

Seleziona l'ennesimo figlio contando dalla fine.

```
tr:nth-last-child(2) {  
    background-color: yellow;  
}
```

:only-child

Seleziona un elemento che è l'unico figlio del suo genitore.

```
div:only-child {  
    border: 1px solid black;  
}
```

:first-of-type

Seleziona il primo elemento di un certo tipo tra i suoi fratelli.

```
h2:first-of-type {  
    text-decoration: underline;  
}
```

:last-of-type

Seleziona l'ultimo elemento di un certo tipo tra i suoi fratelli.

```
p:last-of-type {  
    font-style: italic;  
}
```

`:nth-of-type(n)`

Seleziona il numero `n` di un determinato tipo di elemento.

```
p:nth-of-type(2) {  
  font-size: 18px;  
}
```

`:nth-last-of-type(n)`

Seleziona il numero `n` di un tipo di elemento contando dalla fine.

```
p:nth-last-of-type(1) {  
  text-align: center;  
}
```

`:only-of-type`

Seleziona un elemento se è l'unico di quel tipo tra i suoi fratelli.

```
span:only-of-type {  
  color: orange;  
}
```

:empty

Seleziona gli elementi che non contengono figli (testo o altri elementi).

```
div:empty {  
    display: none;  
}
```

:not(selector)

Esclude un elemento specificato.

```
p:not(.speciale) {  
    color: gray;  
}
```

:checked

Seleziona un `checkbox` o `radio` quando è selezionato.

```
input:checked {  
    outline: 2px solid green;  
}
```

:disabled

Seleziona un elemento disabilitato.

```
button:disabled {  
    opacity: 0.5;  
}
```

:enabled

Seleziona un elemento attivo e non disabilitato.

```
input:enabled {  
    background-color: white;  
}
```

:required

Seleziona i campi obbligatori in un modulo.

```
input:required {  
    border: 2px solid red;  
}
```

:optional

Seleziona i campi che non sono obbligatori.

```
input:optional {  
    border: 1px dashed gray;  
}
```

:read-only

Seleziona i campi di input che sono in sola lettura.

```
input:read-only {  
    background-color: lightgray;  
}
```

`:read-write`

Seleziona i campi di input che possono essere modificati.

```
input:read-write {  
    border-color: blue;  
}
```

`:default`

Seleziona l'elemento predefinito di un modulo.

```
input:default {  
    box-shadow: 0 0 5px blue;  
}
```

`:fullscreen`

Si applica quando un elemento è in modalità a schermo intero.

```
div:fullscreen {  
    background-color: black;  
}
```

::before

Inserisce un contenuto generato prima di un elemento.

```
p::before {  
  content: "--- ";  
  color: red;  
}
```

::after

Inserisce un contenuto generato dopo un elemento.

```
p::after {  
  content: " ✨";  
  color: gold;  
}
```

::first-letter

Applica uno stile alla prima lettera di un elemento di testo.

```
p::first-letter {  
  font-size: 2em;  
  font-weight: bold;  
  color: darkblue;  
}
```


`::first-line`

Applica uno stile alla prima riga di un elemento di testo.

```
p::first-line {  
  font-style: italic;  
  color: gray;  
}
```

`::selection`

Applica uno stile al testo selezionato dall'utente.

```
p::selection {  
  background-color: yellow;  
  color: black;  
}
```

`::placeholder`

Applica uno stile al testo segnaposto di un campo di input.

```
input::placeholder {  
  color: lightgray;  
  font-style: italic;  
}
```

`::marker`

Applica uno stile ai marcatori degli elenchi (``ul`` e ``ol``).

```
li::marker {  
  color: blue;  
  font-size: 1.5em;  
}
```

`::backdrop`

Applica uno stile allo sfondo di un elemento modale a schermo intero. Esempio: si apre una finestra di dialogo, all'interno della quale deve proseguire l'interazione con l'utente finché non viene chiusa; lo sfondo e' lo schermo (intero) che sta dietro, bloccato, a questa finestra modale. "Sei sicuro di voler proseguire?" ...

```
::backdrop {  
  background-color: rgba(0, 0, 0, 0.5);  
}
```

risorse (1/2)

www.w3c.org specifiche del posizionamento : <https://www.w3.org/TR/css3-positioning/> pages
(www.w3.org/TR/css3-values/#viewport-relative-lengths)

comando ultra`css (c'e` anche la versione in italiano ... guarda la parte finale: molti link interessanti ...)
Why **tables** for layout is stupid: problems defined, solutions offered <http://www.hotdesign.com/seibold/everything.html>

una raccolta di CSS properties

(oltre a quelle menzionate nella slide di competenza ... qui some broken links and some tiny explanations)
<http://meiert.com/en/indices/css-properties/>

supporto a css da parte dei browser <http://www.webdevout.net/browser-support-css>
bisogna selezionare (Pick browsers to display) i browser su cui fare il confronto. Ci sono delle guidelines ancora buone da leggere ...

transizioni e animazioni in css3

Esempio di transizione con riferimento alla specifica presente nel W3C:

<http://www.xul.fr/en/css/transition.php>
e temporizzazione: <http://www.the-art-of-web.com/css/timing-function/>

codepip

CSS e altro, giocando.

C'e` una parte free che e` utile e divertente. Riguardo alla parte non free ... beh, intanto ditemi come lo avete usato e poi vediamo ...
<https://codepip.com/> (Avete scoperto cose simili girando per la rete? Ma allora ditemelo!)

formato webP per le immagini

<https://developers.google.com/speed/webp>

risorse con risorse ...

qui si parla di flex ... menzionato a lezione: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
ed in generale <https://css-tricks.com> e` un sito interessante per trovare suggerimenti su soluzioni css

ancora da www.w3c.org

Guidelines sono in http://www.w3.org/TR/xhtml1/guidelines.html#C_09 e http://www.w3.org/TR/xhtml1/guidelines.html#C_14 (C9 e c14 in particolare, ma probabilmente e` meglio dopo aver affrontato XML)

C.9. Character Encoding

Historically, the character encoding of an HTML document is either specified by a web server via the charset parameter of the HTTP Content-Type header, or via a meta element in the document itself. In an XML document, the character encoding of the document is specified on the XML declaration (e.g., `<?xml version="1.0" encoding="EUC-JP"?>`). In order to portably present documents with specific character encodings, the best approach is to ensure that the web server provides the correct headers. If this is not possible, a document that wants to set its character encoding explicitly must include both the XML declaration and an encoding declaration and a meta http-equiv statement (e.g., `<meta http-equiv="Content-type" content="text/html; charset=EUC-JP" />`). In XHTML-conforming user agents, the value of the encoding declaration of the XML declaration takes precedence.

Note: be aware that if a document must include the character encoding declaration in a meta http-equiv statement, that document may always be interpreted by HTTP servers and/or user agents as being of the internet media type defined in that statement. If a document is to be served as multiple media types, the HTTP server must be used to set the encoding of the document.

C.14. Referencing Style Elements when serving as XML

In HTML 4 and XHTML, the style element can be used to define document-internal style rules. In XML, an XML stylesheet declaration is used to define style rules. In order to be compatible with this convention, style elements should have their fragment identifier set using the id attribute, and an XML stylesheet declaration should reference this fragment. For example:

```
<?xml-stylesheet href="W3C-REC.css" type="text/css"?>
<?xml-stylesheet href="#internalStyle" type="text/css"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>An internal stylesheet example</title>
<style type="text/css" id="internalStyle">
  code {
    color: green;
    font-family: monospace;
    font-weight: bold;
  } </style> </head>
<body> <p> This is text that uses our <code>internal stylesheet</code>. </p>
</body>
</html>
```

ancora entities <https://www.freeformatter.com/html-entities.html>

Attività in laboratorio (1/5)

Qui ci sono vari esercizi ... Dopo averne fatti un po', e aver frequentato qualcuno dei link suggeriti nella slide precedente, una fonte di ulteriori esercizi è il seguente **ESERCIZIO GENERICO MA CHALLENGING**:
ci sono vari articoli nel sito <https://www.positioniseverything.net/category/coding/css/>:
costruire una pagina, o un sito web di dimensioni limitate, oppure modificare il
prodotto di un esercizio già svolto, in modo da sperimentare il contenuto di uno o più degli articoli.

Tra gli esercizi della lezione precedente ce ne sono svariati che, una volta risolti solo “strutturalmente”, cioè con XHTML, possono essere estesi con l'uso delle CSS. Tra l'altro viene fuori un risultato più apprezzabile e il lavoro fatto viene valorizzato ... considerate anche quelli ...

(già visto ma non fatto?) XHTML-15

Consultare il tutorial di w3schools (<http://www.w3schools.com/xhtml/default.asp>) non è affatto una cattiva idea; anche fare i test lì presentati è cosa buona.

Inoltre <http://www.w3schools.com/html/default.asp> contiene dei test: provare a farli, anche per verificare cosa potrebbe essere approfondito usando gli esercizi successivi ... Attenti: all'inizio di ogni pagina ci sono i link al “Previous Chapter” e al “Next Chapter”; i medesimi link sono anche alla fine della parte interessante della pagina - il resto, è pubblicità ...

(già visto ma non fatto?) XHTML-CSS-5

Realizzare una pagina web il cui contenuto è arricchito con delle spiegazioni associate a parole/espressioni contenute nel testo. Si tratta di aggiungere dei cosiddetti link di help nella pagina: alcune delle parole o espressioni usate ha vicino una icona di help (un '?' ad esempio) che, in qualche modo, fa accedere a informazioni sulla parola/espressione; si può trattare di una nuova pagina, o una finestra pop up, o una tooltip - nella lezione su xhtml abbiamo visto, ad esempio, come far apparire del testo relativo ad una immagine quando il cursore del mouse va sopra l'immagine (hover, hover, che vuol dire?). Questo non è l'unico ma è un modo. Come fare queste cose va cercato (non chiesto semplicemente ...), senno' l'es. è inutile. Un posto dove cercare, oltre ad altri siti suggeriti prima e dopo di qui, potrebbe essere inizialmente <http://www.ericmeyeronclick.com/bonus/proj04-excerpt.html>

Attività in laboratorio (2/5)

CSS-1

dopo aver analizzato la prima parte della sezione sulle CCS (fino a box model?)

- validare box.html (ci dovrebbero essere delle warning per entities)
- validare externalStylesheet.html (la codifica dei caratteri si può correggere seguendo l'esempio di altri file visti a lezione)
- validare img.html

CSS-2

Cercare informazioni a partire da www.w3.org/Style/CSS/current-work su marquee. Sperimentare marquee in un piccolo esempio (w3schools può aiutare). E poi usarlo in qualche esercizio successivo, in cui il testo da proporre in visione è più corposo.

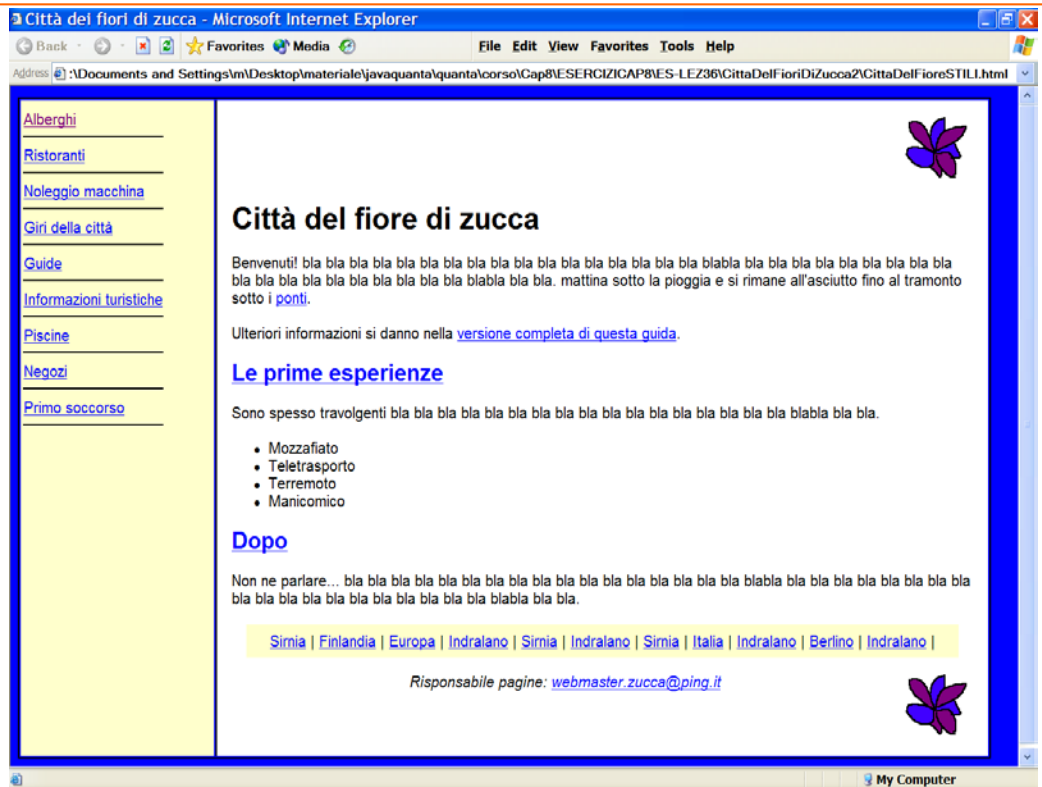
CSS-3

dopo aver analizzato la prima parte della sezione sulle CCS (fino a box model) scrivere un doc xhtml **eser.css.3.html** che riprenda e migliori quello dell'esercizio xhtml sulla città dei fiori di zucca.

Usare comandi di stile **internal**. Per posizionare le immagini usare una regola di stile (con float).

CSS-4

scrivere **eser.css.2.html** che ottenga gli stessi risultati grafici (+/-, licenze poetiche escluse) del precedente, ma usando **internal** o **external** stylesheet.



Attività in laboratorio (3/5)

CSS-5 ... Se non già fatto, sperimentare, in un file `eser.css.5.html`, l'uso di `float` e `clear` per la disposizione di elementi. Posizionare blocchi di testo e immagini dentro al testo.

CSS-6 fare una pagina contenente una tabella in cui appaiono celle di colori diversi (usare i `named colors` e anche colori della palette `websafe`, in modo che le celle contengano colori progressivi). Usare regole di stile `inline` per governare la resa delle celle. La visualizzazione di un colore appare diversa in dipendenza dei colori che gli sono vicini?

CSS-7 sperimentare con la costruzione di liste: fare una pagina contenente i diversi tipi di liste noti. Definire (e discutere nel testo incluso nel documento) diverse versioni delle liste, in modo da variare anche le proprietà `list-style-type` tra i valori possibili. Usare regole di stile incluse in un file esterno. Definire classi per ciascun tipo di lista e sua variante utilizzati. Sperimentare, una o più volte, anche l'uso di `id`, per scrivere una regola che si applicherà ad un solo elemento nel documento. Sperimentare `list-style-image` e `list-style-position`.

CSS-8 produrre un file **`eser.css.8.html`** che spieghi la definizione, gli scopi ed esemplifichi l'uso di almeno tre proprietà. Per i concetti fare riferimento alla definizione CSS 2.1 W3C. Ci si può mantenere ad un livello informale: non è espressamente richiesto di scendere nei particolari del contenuto della specifica. Fare uso di un file di stile **`eser.css.8.css`** appositamente preparato

Attività in laboratorio (4/5)

CSS-9 (liste stilizzate per ottenere un menu`)

Stilizzare una lista in modo che l'elenco fornito come `` diventi un menu` con un proprio sfondo, gestione del link nel menu` diversa da quella nel documento (sperimentando le pseudo-classi di a) e punto elenco costituito da un'immagine. La pagina web del corso costituisce n flebile esempio ...

CSS-10 (positioning)

Scrivere una pagina in cui c'e` solo, iniziando dall'inizio della pagina, una piccola frase in cui una parola e` posizionata in modo assoluto 100 pixel piu` giu` e 50 a destra; cosa succede se il positioning invece e` relativo? Ripetere il medesimo esercizio con una frase disposta in un paragrafo successivo al primo

CSS-11 Supponendo che nel foglio di stile agiscano le seguenti regole:

```
h1 {font-size: 24px;}      h2 {font-size: 18px;}      p {font-size: 12px;}
quali sono i valori di margin-left per h1, h2 e p, dovuti alla ulteriore regola
h1, h2, p {margin-left: 1em;}
```

Pensarci, esprimere la risposta in termini di pixel e verificare, ad esempio scrivendo un documento xhtml contenente gli elementi interessati dalle regole, es.

```
<h1>Left margin = ... pixels</h1> <h2>Lft mrg = ... pxs</h2> <p>Lftmrg = .pxs</p>
```

CSS-12 (layout mediante tabelle e positioning) Qui ci sono metodi di layout: fare una pagina web in un paio almeno di questi metodi, mettendo in evidenza (nella pagina stessa) le caratteristiche che si pensano rilevanti per i (plurale) layout scelti

- The Layout Reservoir: <http://www.bluerobot.com/web/layouts/>
- boxes a go go: www.thenoodleincident.com/tutorials/box_lesson/boxes.html
- molti interessanti layout: www.meyerweb.com/eric/css/edge/

- [incluso solo per la giraffe ;)] alcuni di questi layout tengono anche conto della necessita` di fare cose compatibili con browser piu` datati: www.realworldstyle.com/

Attività in laboratorio (5/5)

CSS-13 (layout mediante tabelle e positioning)

Realizzare una pagina in cui si discute e si esemplifica l'uso delle css per definire il layout dei documenti html. Ci si aspetta che si discuta almeno il layout basato su table e quello basato su uso di div.

Ci sono esempi nel materiale visto o accennato a lezione. Inoltre per realizzare il layout si puo` cercare informazioni nel sito <http://www.alistapart.com/> o

<http://www.w3schools.com/css/default.asp>, come riferimenti iniziali.

Inutile applicarsi a cose complicate (almeno in questo esercizio).

Un classico layout a tre colonne e` perfettamente sufficiente (cioe` il *santo graal* e` sufficiente: vedi <http://www.alistapart.com/articles/holygrail>).

CSS-14 (varie possibili scelte per un sito web di documentazione)

Produrre un minisito web, in cui siano realizzati alcuni dei seguenti compiti

- Spiegazione della differenza tra elementi block level e inline;
- Spiegazione del significato dell'espressione "documento valido" e di come si puo` verificare la validita` di un documento;
- Spiegazione del significato dell'espressione "documento well-formed" e di come si puo` verificare la well-formedness di un documento;
- Spiegazione di a cosa serve una "document type description" discutendo esempi a riguardo
- che cosa e` un linguaggio di markup; menzionare diversi linguaggi di markup; fare esempi relativi ad almeno un linguaggio di markup conosciuto ...
- Descrizione dei concetti di elemento e di attributo in XHTML, con opportuni esempi;
- Spiegazione della definizione slide8-lezione1: si riconoscono elementi? E attributi?
- Gestione di immagini di background; spiegazione di come e dove si e` trovata la (le) proprieta` css utilizzata(e); spiegazione basata sulla documentazione di quali alternative sono possibili nella gestione dell'immagine di background;
- Uso di opportune proprieta` in regole css per ottenere un layout a tre colonne (holy grail); utilizzare i siti web consigliati e la documentazione per trovare le proprieta` in questione; integrare nelle pagine spiegazioni sulle caratteristiche di tali proprieta` - menzionare la fonte da cui tali informazioni sono state tratte (no_consigli_di_amici ...)
- Uso e spiegazione di entita`; cosa sono le entita`? Fare esempi di entita`, mostrandole e definendole brevemente; spiegare per ciascuna da dove, nella documentazione o su testi, sono state tratte (il tutto all'interno della pagina web).

fixed.menu.html

```
A:link {color:yellow}
A:visited {color:#CCCC33}
A:active {color:yellow}
A:hover { color: #999900 }
```

```
.testo { position: absolute;
top:0%;
left:20%;
}
```

```
.menu { position:fixed;
top:15%; left:2%;
width: 10%; ...}
```

```
.titolo {
position: fixed;
top:1%; left:2%;
background: black;
...}
```

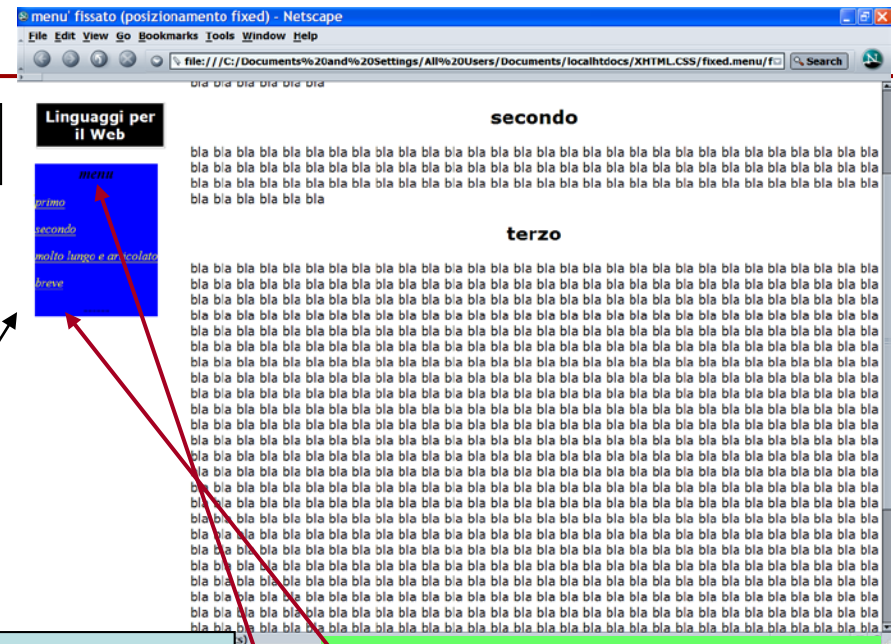
```
... <p class="titolo">Linguaggi per il Web</p>
<div class="menu">
<p><a href="#primo">primo</a></p>
<p><a href="#secondo">secondo</a></p>
<p><a href="#terzo">molto lungo e articolato</a></p>
<p><a href="#quarto">breve</a></p>
</div>
```

```
<div class="testo">
<h2 id="primo">primo pr
```

fixed.menu.css

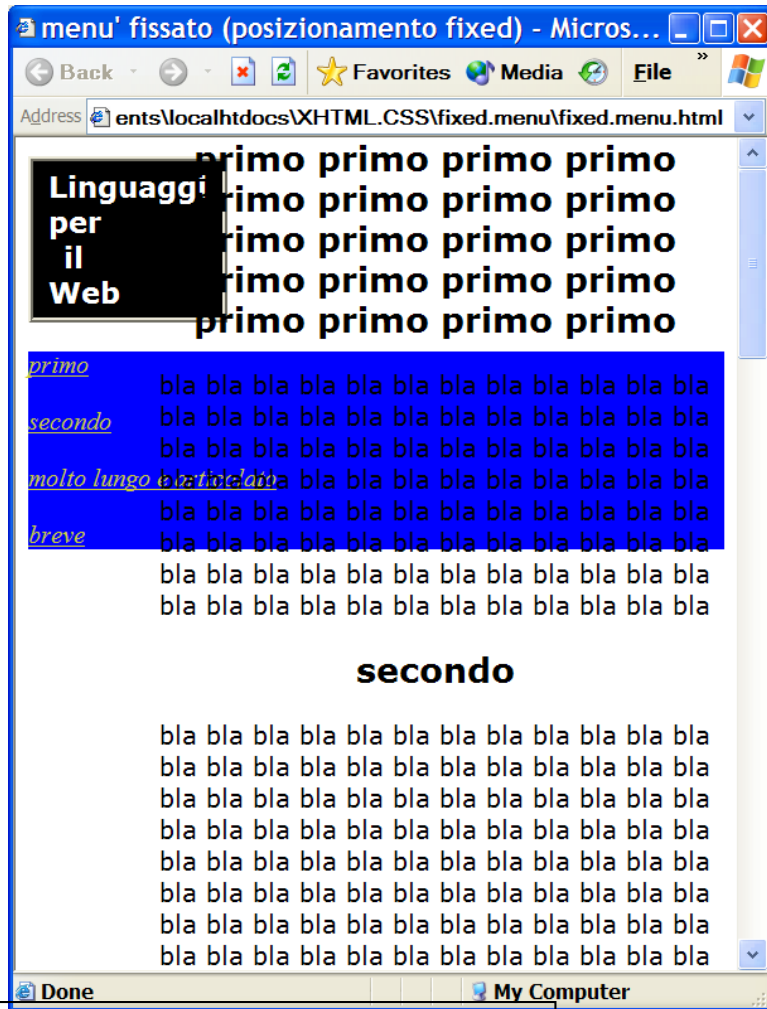
```
<div class="browserWarning">
** ... **</div>
nel doc non verra' visualizzato
```

Indirizzamento di frammenti del doc:
#primo riferisce all'elemento avente attr.
id="primo" (che deve essere unico)

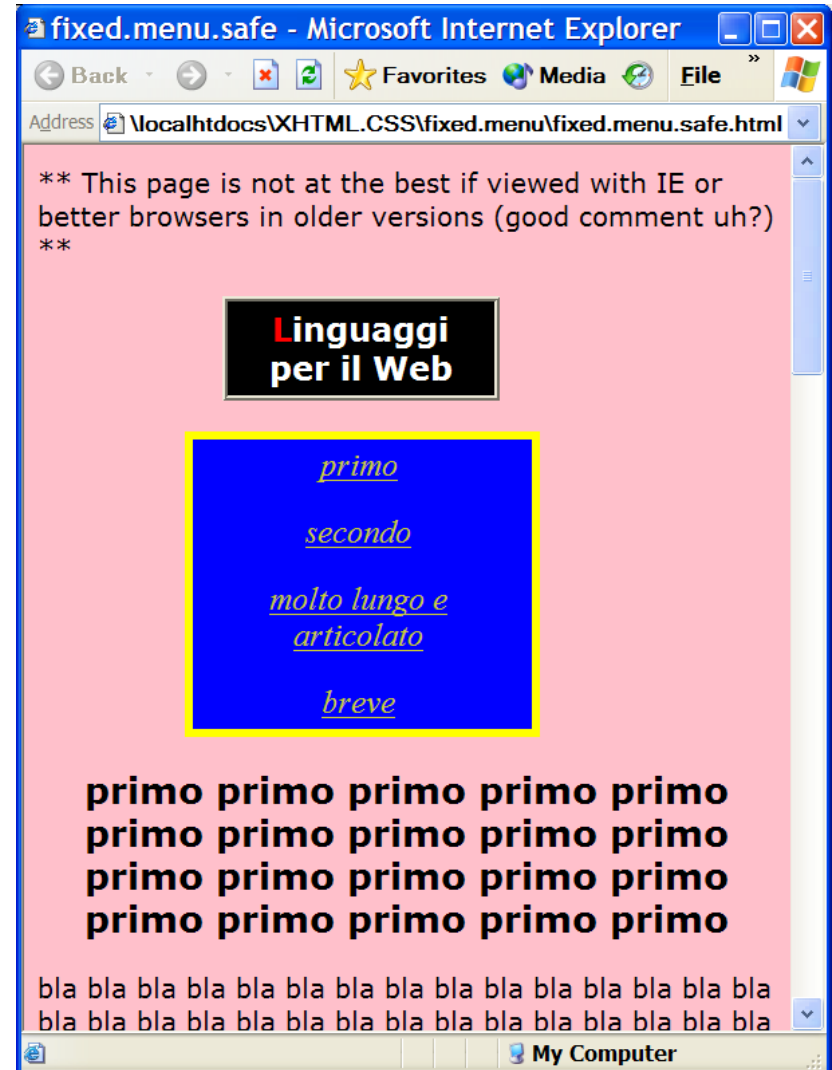


```
.menu:after {
content: "-----";
display: block;
text-align: center;
color: black;}
.menu:before {
content: "menu";
display: block;
text-align: center;
color: black;
font-weight: bold;
font-size: 120%;}
.browserWarning
{display:none}
```

Vintage ... fixed.menu.html (problemi con IE fino a 7*)



(vecchio IE (non piu!)
presente in lab)



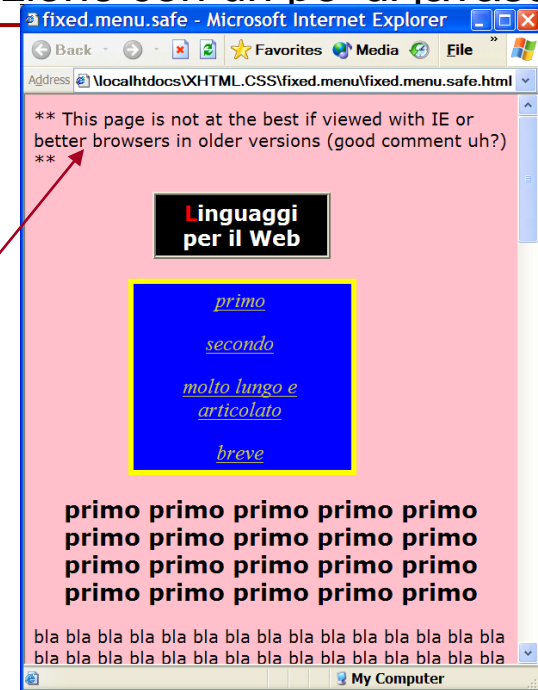
In questi casi meglio cambiare stile per i browser che non supportano bene queste peculiarita': invece dello stile fixed.menu.css (sinistra), usare dummy.fixed.menu.css per IE 6,7 (destra)

```
.testo {}
.menu {border: solid thick;
padding:.5ex .5em; text-align:center;
width: 60%; margin-left: 20%;
background: blue; color: yellow;
font-family: Times New Roman, serif;
font-style: oblique; font-size: 120%}
.titolo {
width: 50%; margin-left: 25%;
...}

.titolo:first-letter {color: #FF0000}

/* .browserWarning {display:none} */
```

dummy.fixed.menu.css



```
<head>...
<script type="text/javascript">
<!--
// This style sheet works for IE 4+ and Netscape 6
if (navigator.appName == "Microsoft Internet Explorer") {
document.write("<link rel='stylesheet' href='fixed.dummy.menu.css'>")
else document.write("<link rel='stylesheet' href='fixed.menu.css'>")
// -->
</script>      <!--<link rel="stylesheet" href="fixed.menu.css" />-->
</head><body>
<div class="browserWarning">
** This page is not at the best if ...(good comment uh?) **</div>
<p class="titolo">Linguaggi per il Web</p>
```