Internet Software Technologies
CSS

IMCNE

A.A. 2008/09

Gabriele Cecchetti

Why CSS (1/2)

- Divide contents from styles.
- Common styles inside one external style sheet (site's styles sheet)

Why CSS (2/2)

- CSS are are better than HTML for:
  - **text management**
  - **backgrounds**
  - **margins e borders**
  - **Layouts**
  - **Medim dependent style**

How link CSS to HTML

- Two main types of sheets: *internal* and *external.*

Inner Style sheet

- When CSS code is inside HTML document.
- 2 different ways to insert an inner style sheet:
  - **inline** or
  - **inside heading of document**.

Inline style sheet

```
<P style="color: red; margin-left:
  10%">
  This text is formatted by inline style
  information</P>
```

Inside style sheet

```
<html><head>
<title>Fogli di stile
  incorporati</title>
<STYLE type="text/css">
<!--
body { background: #aaaaaa }
-->
</STYLE>
</head>
<body>...
```


STYLE

- **STYLE** has 2 attributes:
  - **`type`** (mandatory): for identify incorporated data type; only one value : `text/css`;
  - **`media`** (optional): medium where style sheet is applied (screen, printer …).

External style sheet

- Usually inside a text file with `.css` extension.
- 2 ways to attach an external style sheet.

External Style sheet: LINK

- 1st metod <LINK>:

```
<html>
<head>
<title>External style sheet</title>
<LINK rel="stylesheet" href="stile.css"
  type="text/css" media="screen">
</head>
<body> ...
```

External Styel Sheet:

- 2nd method:

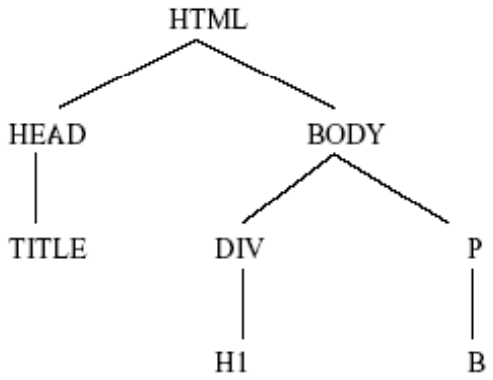
```
<head>
<style>
<!-- @import url("stile.css"); -->
</style>
```


Which method is better ?

- Common styles inside one external style sheet => Next we can modify just this file.
- Then we can establish if somewhere, for some pages need some changes:
  we may use @import to overwrite generic style sheet.
- Generally is better do not use inside style sheet because of reusability.

Document Tree Structure

- HTML document has a tree structure like the following:



Inherithance

- If we define a style for an element, descendants of this element will receive the same style, except if this element is defined again.

Inherithance: an Example

```
<head>
<style type="text/css">
body { color: red }
h1 { color: black }
</style>
</head>
<body>
Document text (red)
<p>Paragraph <b>text</b> (inherit color) </p>
<h1>Here the style is redefined</h1>
</body>
```


Cascading

- Why “*Cascading*” ?
- Let’s see browser’s operations to choose sheets to load.

Browser style operations to display a document

- Check media attribute
- Verify sheet origin; 3 types:
  - Author's sheet;
  - User's sheet;
  - Browser's sheet.
- Choose style for each element, using selector specificity: (id, classes, elements);
- If more styles refer the same element, browser look at document structure (inline styles are considered before inside styles, and these before external styles).

Exemple 1:

```
1. p { background: yellow; color: red }
   // external sheet
2. p { background: black; color: white }
   //
3. <p style="background: red; color:
   black">
```

Example 2:

```
<head>
<style type="text/css">
p { color: blue; }
.green { color: green; }
</style>
</head>
<body>
<p>Text paragraph</p>
<p class="green">other text</p>
</body>
```


Medium dependent styles

- **media**, attribute
- directive @**media**
- The following media are valid:
  - **all** (default), **aural**, **braille**, **embossed**, **handheld**, **print**, **projection**, **screen**, **tty**, **tv**

Example

```
<style type="text/css" media="screen">
h1 { background:black; color:white; text-align:center }
h2 { color: red; font-style:italic}
h3 { display: none }
</style>
<style type="text/css" media="print">
h1 { color: black; text-align: left }
h2 { display: none }
</style>
<body>
<h1>Test Printer Preview</h1>
<h2>This is not printed</h2>
<h3>This is not displayed on screen</h3>
</body>
```


CSS Rules

- CSS rule is composed by 2 parts: one **selector** and one for **declarations**.

  SELECTOR { property: value; property: value}

Example: CSS rules

```
H1 {
  text-weight: bold;
  text-align: center;
  color: blue
}
```


Comments

- C syntax:

```
/* begin comment;
end comment */
```

Selectors

- Elements' proprieties are assigned by selectors:

```
p { text-align: justify }
div {
  position: absolute;
  left: 10px;
  top: 50px
  }
table { width: 80%; height: 50% }
```


Grouping selectors

```
h1 { font-family: serif }
h2 { font-family: serif }
h3 { font-family: serif }
```

- It's the same:

h1, h2, h3 { font-family: serif }

Universal selector

```
<style type="text/css">
body { font-size: large }
* { color: red; }
</style>
<body>
This is red <p> and this </p>
<blockquote> and this too </blockquote>
</body>
```


Class selectors

```
P.classname1 { … }
P.classname2 { … }
```

- Use:

```
<P class="classname1">…</>
<P class="classname2">…</>
```

Anonymous classes

```
.classname {…}
```

- Use:

```
<P class=“classname1”>…</>
<h3 class=“classname1”>…</>
```


ID Selector

- ID attribute identify **one** element.
- Useful for scripting languages to identify the elements.

Example: ID selector

```
#idname { … }
```

- Use:
  `<div id="idname"> … </div>`

PseudoClassess: examples

```
a:link { color: blue; }
a:visited { text-decoration: none }
a:hover { text-transform: uppercase }
a:active { color: red }
```

PseudoElements: examples

```
p:first-letter { font-size: xx-large }
p:first-line { font-style: oblique }
```


Selectors, tree structure and attributes

Syntax example	Use it when
div p	P is descendant of DIV element
div > p	P è ***child*** of DIV elementa
div + p	P ***follows immediately*** DIV elem.
p[attr]	P has set attribute attr
p[attr="value"]	P has attribute attr ***equal to*** "value"
p[attr~="value"]	P has attribute attr ***containing*** "value"
p[attr|="value"]	P has attribute attr ***which begin for*** "value"

Colors and backgrounds

- *foreground color*
- *background color*
- *border color*

Foreground color: color

- Color names:
  p { **color: *red*;** }
- Hexadecimal Codes
  .class1 { **color**: ***#00CC00*** }
- Functional notation
  .class1 { **color: rgb*(0, 204, 0)*** }
  .class1 { **color: rgb*(0%, 80%, 0%)*** }

Background: examples

```
body {
   background-color: #ccc }
   background-image: url("ball.gif");
   background-repeat: repeat[-x,-y]
                          or  no-repeat
   background-position: 50px 100px
}
```


Dimensions

- **in** (**inches**), **cm** (**centimeters**), **mm** (**millimeters**)
- **pt** (**points**): = 1/72 of inch;
- **pc** (**picas**): =12 points;
- **em**: = medium height of a character for a certain font;
- **ex**: height of 'x';
- **px** (**pixel**)

Text formatting (1/2)

```
p {
   text-align: [left | right | center | justify ];
   text-indent: 10px;
   text-decoration: [none | underline | overline |
      line-through | blink ];
   text-transform: [ none | capitalize | uppercase]
}
```


Text formatting (2/2)

```
P {
   letter-spacing: -2px;
   word-spacing: 25px;
   white-space: [normal | pre | nowrap];
}
```

Font styles

- Choose font
- Choose font's property

Font-family

```
p { font-family: "Algerian", "Times New Roman", serif }
```

- The last is a generic family, like:
  - **serif**:
  - **sans-serif**:
  - **cursive**:
  - **fantasy**:
  - **monospace**:

Font-size

- Use numeric values (using dimension units) or percentual size of parent element. Example:

```
h2 { font-size: 30pt }
```

- Use by keywords: **xx-small**, **x-small**, **small**, **medium** (default), **large**, **x-large**, **xx-large**. Example:

```
p { font-size: large }
```

- Use relative values: **smaller**, **larger**.

Font-style

- *normal*, *italic* e *oblique*. Example:

```
p { font-style: italic }
```

Font-weight

- Boldness…
- Numeric value range from 100 to 900 with step 100 (100, 200, 300, 400, 500, 600, 700, 800 and 900), greater values for bolder ones
- Keywords: **normal** (400), **bold** (700)
- Relative keywords:**bolder**, **lighter**.
- Example:

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


Font-variant

- One value: **smallcaps**;
- Example:

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

Font-stretch

- Width of font… possible values:
  - **ultra-condensed**,
  - **extra-condensed**,
  - **condensed**,
  - **semi-condensed**,
  - **normal**,
  - **semi-expanded**,
  - **expanded**,
  - **extra-expanded**,
  - **ultra-expanded**.
- Relative keywords: **wider** e **narrower**

Line-height

- Examples:

```
.par1 { font-size: 12pt; line-height: 150% }
.par2 { font-size: 12pt; line-height: 0.6 }
```

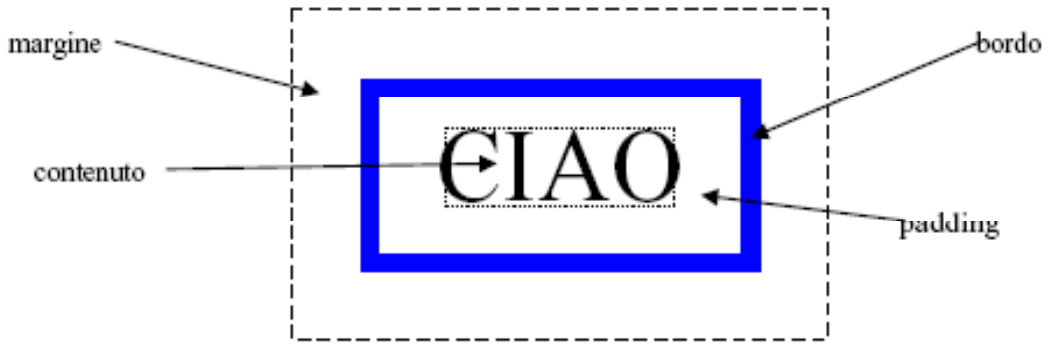
Font (single property)

- **One rule** for **font-weight**, **font-style**, **font-variant**, **font-size/line-height** e **font-family**.
  Example:

```
#myfont {
font: bold italic 12pt/20pt
  "Helvetica",serif
}
```


Box Model

- Consider<P>CIAO</P>
- CSS:



Example

```
<style type="text/css">
body { margin: 0 }
div {
width: 200px;
margin: 20px 50px;
padding: 15px;
border: thick dashed blue;
background: yellow;
}
</style>
…
<body>
<div>Test block to learn CSS</div>
<div>Test block to learn CSS</div>
</body>
```


Width and height of content area

- Example:

p { **width: 200px; height: auto;** }

padding

- padding-top
- padding-bottom
- padding-left
- padding-right

border-color

- border-top-color
- border-bottom-color
- border-left-color
- border-right-color

border-width

- **border-top-width**
- **border-bottom-width**
- **border-left-width**
- **border-right-width**

border-width example

```
div {
    border-top-width: thin;
    border-right-width: medium;
    border-bottom-width: thick;
    border-left-width: 5px;
}
```

- or…

div { **border-width: *thin medium thick* 5px** }

border-style

- border-top-style
- border-bottom-style
- border-left-style
- border-right-style

border-style (value)

- none
- hidden
- dotted
- dashed
- solid
- double
- groove
- ridge
- inset
- outset

margin

- **margin-top**
- **margin-bottom**
- **margin-left**
- **margin-right**

position

- Values:
  - **static**
  - **absolute**
  - **relative**
  - **fixed**
- Examples:
  - img { **position: relative; left: 50px** }
  - img { **position: absolute; left: 400px;**
    top: 100px }
  - img { **position: fixed**; left: 400px; top: 30px }

z-index … by example (1/2)

```
#padre1 {
position: absolute;
left: 10px; top: 10px;
width: 300px; height: 200px;
background-color: yellow;
z-index: 3;
}
#padre2 {
position: absolute;
left: 50px; top: 50px;
width: 300px; height: 200px;
background-color: red;
z-index: 2;
}
```


z-index … by example (2/2)

```
#figlio1 { position: absolute; z-index: 1}
#figlio2 { position: absolute; z-index: 4}
<body>
<div id="padre1">
Padre 1
<p id="figlio1">Figlio1</p>
</div>
<div id="padre2">
Padre 2
<p id="figlio2">Figlio2</p>
</div>
</body>
```

float

- Values:
  - **none**
  - **left**
  - **right**

clear

- **Values:**
  - **none**
  - **left**
  - **right**
  - **both**

visibility

- Values:
  - visible
  - hidden
  - collapse

overflow

- Values:
  - hidden
  - scroll
  - auto

cursor

- **Values:**
  - **auto**
  - **crosshair**
  - **default**
  - **pointer**
  - **move**
  - ***-resize**
  - **text**
  - **wait**
  - **help**

display

- **Values:**
  - **block**
  - **inline**
  - **list-item**
  - **marker**
  - **none**
  - **run-in** and **compact**
  - **table**, **inline-table**, **table-row-group**, **table-column**, **tablecolumn-group**, **table-header-group**, **table-footer-group**, **tablerow**, **table-cell**, **table-caption**