Introduction to Web Programming

Dott. Salvatore Alaimo

Studio 32 – Blocco 2

E-Mail: alaimos@dmi.unict.it Web: http://www.dmi.unict.it/~alaimos/

Overview

1. HTML & CSS
    a) Introduzione
    b) Alcuni tag HTML
    c) CSS
2. PHP
    a) Introduzione
    b) La sintassi

Per approfondire e Risorse

- http://www.slideshare.net/julieiskander1/html-and-css-part-1
- http://www.slideshare.net/julieiskander1/html-and-css-part-2
- http://www.slideshare.net/julieiskander1/html-and-css-part-3
- http://php.net/manual/en/
- http://pecl.php.net
- http://pear.php.net
- http://www.phpclasses.org
- Libri:
  - **Html5: The Definitive Guide**. Chuck Musciano, Bill Kennedy, Estelle Weyl. O'reilly.
  - **CSS: The Definitive Guide**. Eric A. Meyer. O'reilly.
  - **Javascript: The Definitive Guide**. David Flanagan. O'reilly.
  - **Programming PHP**. Kevin Tatroe, Peter MacIntyre, Rasmus Lerdorf. O'Reilly.
  - **Learning PHP 5**. David Sklar. O'Reilly.

Parte I

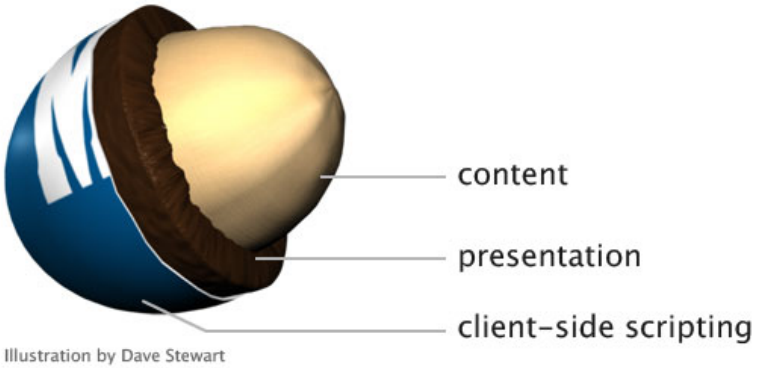
HTML & CSS

Parte Ia

Introduzione

Cosa sono HTML e CSS

- **HTML** fornisce informazioni sulla struttura dei documenti nel web:
  - Dice al browser qual è il contenuto
- **CSS** fornisce informazioni su come presentare gli elementi di una pagina web:
  - Dice al browser come appare la pagina
- **HTML**:
  - Non è un linguaggio di programmazione ma un linguaggio di **MARKUP**
    - Usa **tag** e **keywords** per definire il contenuto di una pagina.
  - Sono semplici file di testo con estensione .html che possono essere modificati con un semplice editor di testo.
  - Non ci sono informazioni sullo stile da usare per visualizzare gli elementi.



Standards

- …indicano allo sviluppatore quali tag e keywords usare
- …indicano al browser cosa fare con i file HTML:
  - **HTML**: vecchio non usare
  - **xHTML**: attualmente il più supportato
  - **HTML5**: nuovo in fase di affermazione come nuovo standard principale

Standards

- Nonostante gli standard…i browser non lavorano allo stesso modo!
- Quindi? Si possono usare i validatori per accertarsi di rispettare gli standard:
  - HTML: http://validator.w3.org ; CSS: http://jigsaw.w3.org
- Ma questo non è sempre sufficiente
  - Soluzioni?
    - Testare tutto su più browser (Firefox, Chrome, Safari, IE)
    - Browsershots.org

Standards

- I **tag** in un documento HTML sono tipicamente racchiusi tra **parentesi angolari** (< e >)
  - es: , <!-- Un Commento-->
- Normalmente essi sono **sempre accoppiati**:
  - ogni tag ha un **tag di chiusura** (es: …)
  - alcuni tag non richiedono un tag di chiusura (es:
, <img>)
- I tag possono anche contenere **attributi**:
  - hanno la forma **nomeAttributo="valore attributo"**
  - es: ….
- Ogni documento HTML ha un insieme di **tag standard** che devono essere indicati sempre e definiscono la struttura base del documento.

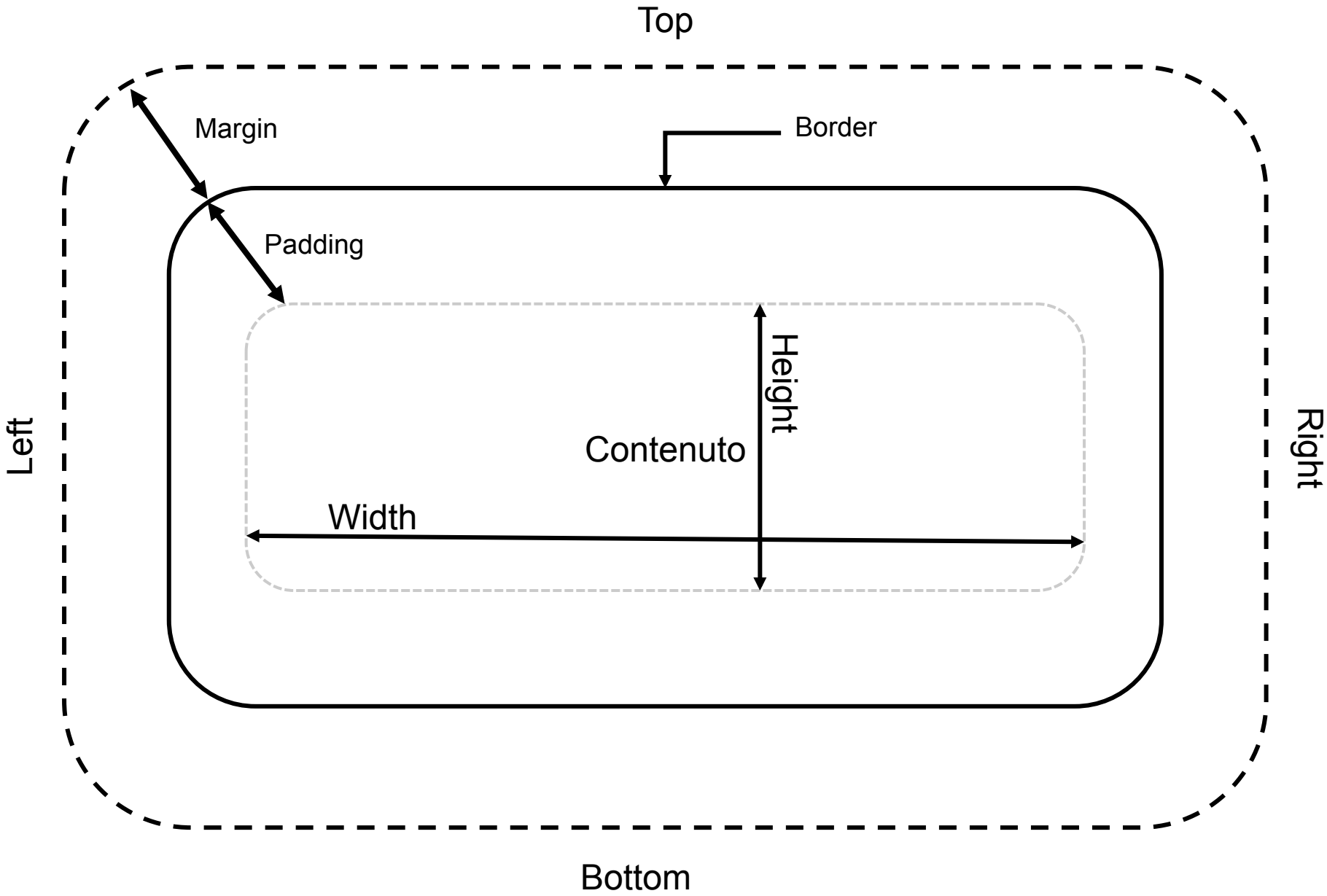
Standards

Head

Body

```
<!DOCTYPE html>
<html lang="en-US">
        <head>
                <title>Display demo</title>
                <meta name="description" content="A demo page to show HTML
display property">
                <link rel="stylesheet" type="text/css" href="style.css">
                <script type="text/javascript">
                        // js code goes here
                </script>
        </head>
        <body>
                <div id="main_page">Welcome to the unlimited world of
                HTML </div>
        </body>
</html>
```

Box Model

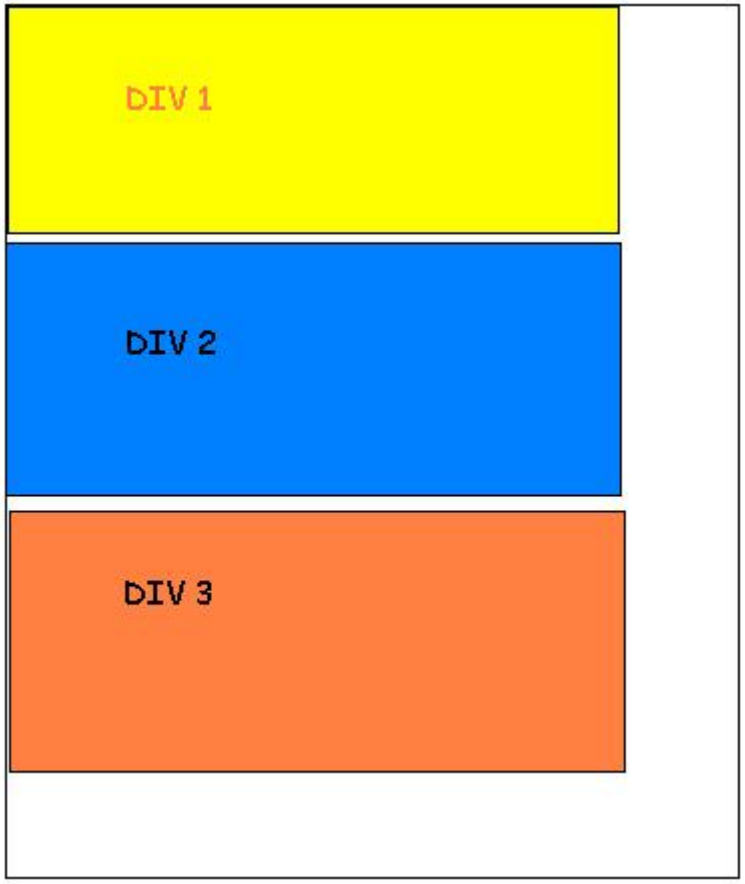

Top
Margin
Border
Padding
Height
Contenuto
Width
Left
Right
Bottom

Display

- **none**
- **block**
- inline-table
- table-row-group
- table-footer-group
- table-column-group
- table-cell
- **inline**
- **list-item**
- **table**
- inline-block
- table-header-group
- table-row
- table-column
- table-caption

Block Elements

Gli elementi in blocco (, <p>,…) sono mostrati dal browser uno sopra l'altro.




```
<body>
        <div id="div1"></div>
        <div id="div2"></div>
        <div id="div3"></div>
</body>
```

Inline Elements

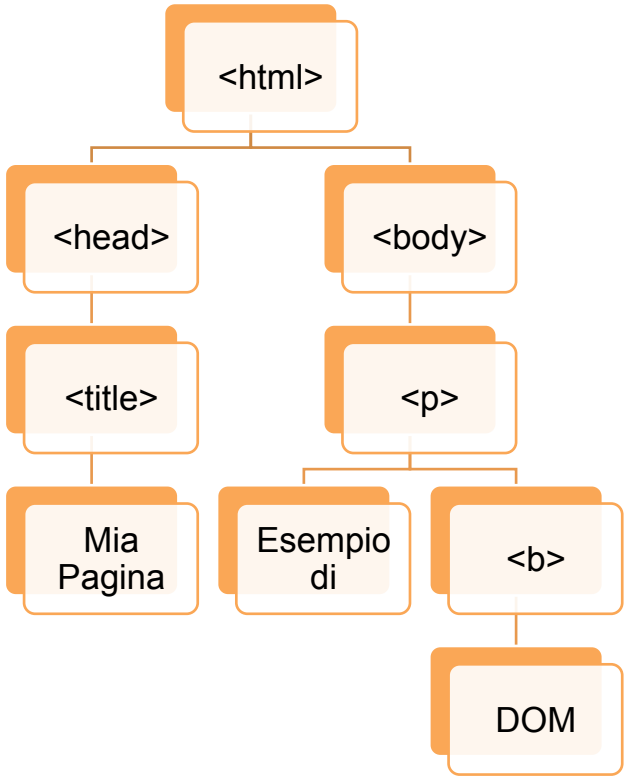
Gli elementi inline (<a>,,<img>,,…) sono mostrati dal browser uno accanto all'altro e appaiono in una nuova riga solo se non c'è abbastanza spazio per mostrare un altro tag


```
<div id="row1" >
    <span class="norm">This is small text and </span>
    <span class="big">this is big</span>
    <span class="italicText"> I am Italic</span>
</div>
```

HTML Document Object Model (DOM)

- Cosa fa un browser:
  - legge un documento HTML (**parsing**)
  - visualizza documento (**rendering**)
- Nella fase di parsing il browser
  - legge i tag presenti nel documento
  - li suddivide in tutte le componenti
  - costruisce un albero che ne rappresenta la struttura (**DOM**)
- Nell'albero ogni nodo è un elemento (tag o testo) contenuto nel file HTML.
- La radice è sempre il tag **<html>**
- I figli di un nodo sono i tag o testi contenuti in esso.

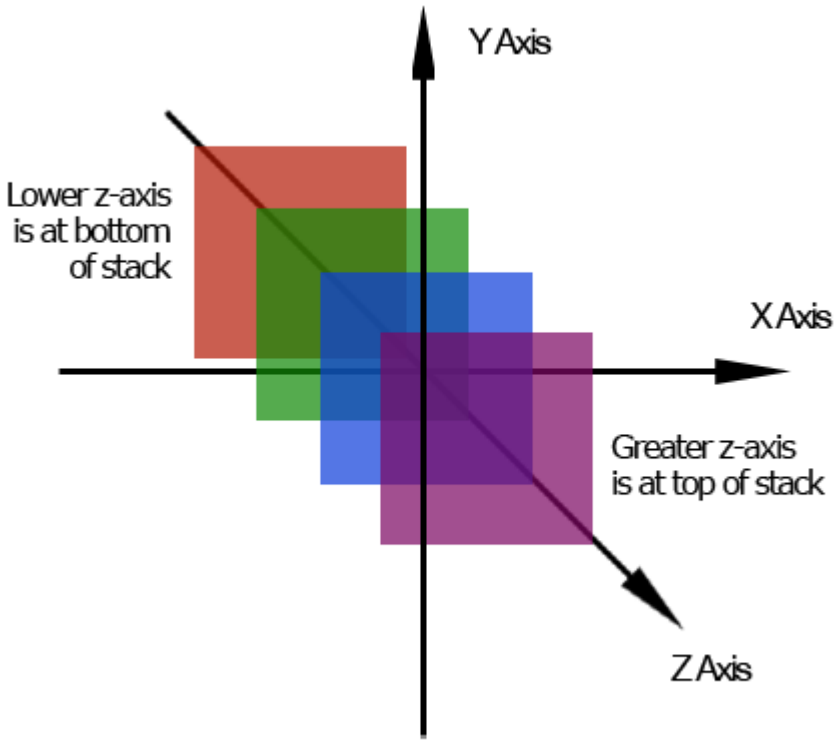
```
<html>
   <head>
      <title>Mia pagina</title>
   </head>
   <body>
      <p>
         Esempio di <b>DOM </b>
      </p>
   </body>
</html>
```



Flusso 3D di una pagina HTML

- Il contenuto di una pagina HTML può essere immaginato in una **struttura tridimensionale**.
- La posizione sull'asse z di un elemento della pagina (**z-index**) corrisponde al numero di tag in cui esso è contenuto.

```
<html>
  <body>
    <p>
      <strong>Hello</strong>
    </p>
  </body>
</html>
```

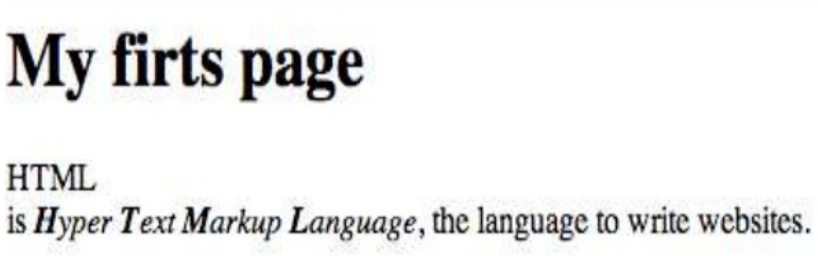


Parte Ib

Alcuni TAG HTML

Un primo semplice esempio

```
<!DOCTYPE html>
<html>
    <head>
        <title>My First Page</title>
        <meta lang="en" />
    </head>
    <body>
        <h1>My firts page</h1>
        <p>HTML <br>
        is <em> <b>H</b>yper <b>T</b>ext <b>M</b>arkup
        <b>L</b>anguage</em>, the language to write websites.
        </p>
    </body>
</html>
```



- Tutti gli **spazi/tab/nuove righe** sono interpretati come un **singolo spazio**.
- Per andare a capo usare il tag **
**
- Per aggiungere più di uno spazio usare il carattere di escape ** **
- Per una lista dei caratteri di escape:
  - http://www.theukwebdesigncompany.com/articles/entity-escape-characters.php

Immagini, Ancore e Link

- Per le immagini si usa il tag **<img>**
- **<img src="url" alt="text">**
- **alt** è il testo che appare se l'immagine non può essere caricata.
- Esempi
  - <img src="http://www.example.com/image.gif" alt="Example Image">
  - <img src="image.gif" alt="Example Image">

- Un **link** è creato tramite il tag **<a>**
- **Contenuto**
- Esempi:
  - Vai a google
  - Libri preferiti
  - Email
- **Ancore**: sono dei punti speciali di una pagina a cui può essere fatto riferimento.
- Esempio:
  - book1.html
    -
  - index.html
    - Capitolo 1

Tabelle

```
<table border="2">
        <tr>
        <th>Heading 1</th><th>Heading 2</th><th>Heading 3</th>
        </tr>
        <tr><td colspan="3">Row 1 cell 1 and 2 and 3 </td></tr>
        <tr>
                <td rowspan="2">Row 2 and 3 cell 1</td>
                <td>Row 2 cell 2</td>
                <td>Row 2 cell 3</td>
        </tr>
        <tr>
                <td>Row 3 cell 2</td>
                <td>Row 3 cell 3</td>
        </tr>
        <tr>
                <td colspan="3">Row 1 cell 1 and 2 and 3 </td>
        </tr>
</table>
```

Heading 1	Heading 2	Heading 3
Row 1 cell 1 and 2 and 3		
Row 2 and 3 cell 1	Row 2 cell 2	Row 2 cell 3
	Row 3 cell 2	Row 3 cell 3

Parte Ic

css

CSS (Cascading Style Sheet)

- È un linguaggio che descrive il **rendering** di un documento strutturato (HTML o XHTML)
- Fornisce al browser il necessario per **gestire la formattazione e il layout** di un documento HTML
- **Non è un linguaggio di markup**
- «**cascade**» → fogli di stile multipli sono miscelati assieme per formattare una pagina. Se avviene un conflitto l'ultima regola è quella applicata.
- Un foglio di stile è un **file con estensione «css»** contenente un **elenco di regole** per la formattazione degli elementi in una pagina.
- La sintassi generale è:
  - **Selettore** {

    proprietà1: valore1;

    proprietà2: valore2;

    …..

    }

CSS (Cascading Style Sheet)

Esempio:

```
body {
    background-color: blue;
    color: white;
    font-size: 24pt;
}
p {
    color: yellow;
}
```

CSS (Cascading Style Sheet)

- Come si aggiunge ad una pagina HTML?
  - Direttamente su un elemento aggiungendo **l'attributo style**

    ```
    <p style="color: red; font-size: 20pt;"
    ```

  - Nel tag <head> dentro il **tag <style>**

    ```
    <head>
        <style type="text/css">
            p { color: red; }
        </style>
    </head>
    ```

  - In un file esterno con estensione «css»
    - Per collegare una pagina html a un file css si usa il **tag <link>**

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

CSS (Cascading Style Sheet)

- Tipi di selettori:
  - Semplici
  - Di gruppo:
    - Elenco di selettori semplici separati da virgole
- Un selettore semplice può essere:
  - Il nome di un elemento (p, h1, div, span,…)

    p { color: red; }
  - Un attributo:

    [href] { font-size: 20pt; }

    h1[title] { color: red; }

    a[href="http://www.google.com"] { color: blue; }
  - Una classe:

    .maincontent { color: white; }
  - Un ID:

    #maincontent { color: red; }

CSS (Cascading Style Sheet)

- Un selettore semplice può essere:
  - Una pseudoclasse:
    - Link:
      - :link, :visited
    - Azioni:
      - :hover, :focus
  - Una combinazione:

    h1.maincontent { color: red; }

    p#maincontent { color: white; }

    a:hover { text-decoration: underline; }

    a:visited { color: black; }

    h1 em { … }

    ul > li { … }
  - Uno pseudoelemento:
    - ::before, ::after

      p::after { content: "test"; color: red; }

Colori e unità di misura

- I colori possono essere indicati usando:
  - **Parole chiave**: red, black, white
  - **Codice esadecimale**: #FF0000, #000000, #FFFFFF
  - **rgb()**: rgb(255,0,0), rgb(0,0,0), rgb(255,255,255)
- Le principali unità di misura sono:
  - pixel (px)

    p { width: 500px; }
  - percentuale

    p.half { width: 50%; }
  - Punti (pt)

    p { font-size: 20pt; }

Formattazione del testo

- Alcune proprietà per la formattazione del testo:
  - color: un colore;
  - text-align: **left**|right|center|justify;
  - text-decoration: **none**|underline|overline|line-through|blink;
  - text-indent: length|**0px**;
  - text-transform: capitalize|uppercase|lowercase|**none**;
  - font-family: il nome di un font;
  - font-size: dmensione|smaller|larger|xx-small| x-small|small|medium|x-large|xx-large;
  - font-weight: normal|bold|bolder|lighter|100-900
  - font-style: normal|italic|oblique
  - font: font-family font-size font-weight font-style;

Altre proprietà

- display: block|inline|table|inline-block|list-item|….;
- z-index: un valore;
- background-color: un colore;
- background-image: url(path di una immagine)|**none**;
- background-repeat: **no-repeat**|repeat|repeat-x|repeat-y
- width, height, padding, margin, padding-left, padding-right, padding-top, padding-bottom, margin-left, margin-right, margin-top, margin-bottom
- border: size type color;

  type uno tra **none**, hidden, dotted, dashed, solid, double

  es: border: 1px solid red;

Parte II

PHP

Parte IIa

Introduzione

Cosa è PHP

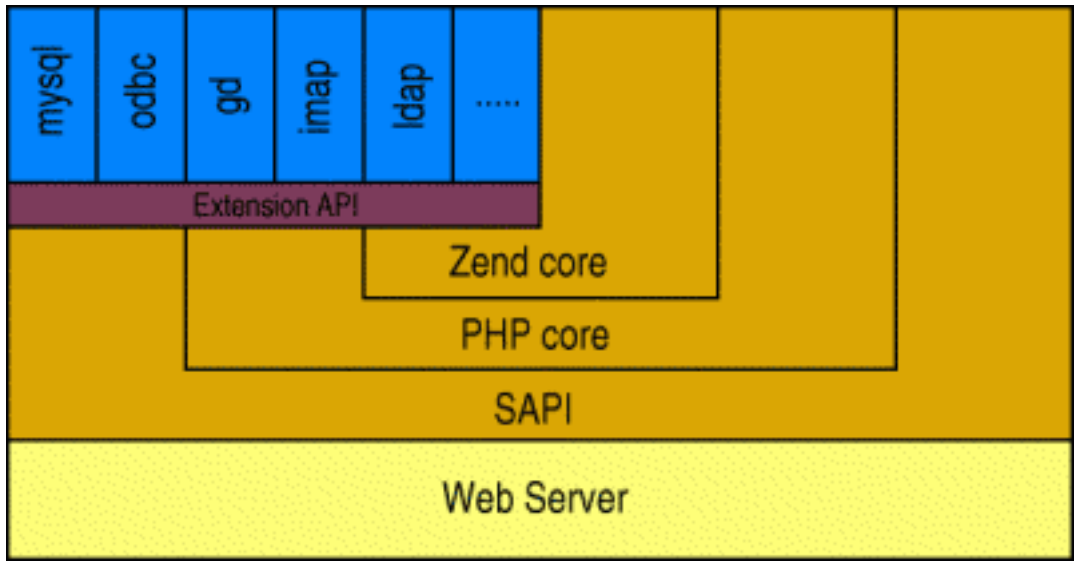
- PHP è un linguaggio di scripting server-side
  - PHP vuol dire «PHP: Hypertext Processor»
  - La sintassi è basata su Perl, Java e C
  - Ad oggi è molto usato per la creazione di contenuti dinamici per via della sua potenza e velocità

Un po’ di storia

- Iniziato come una estensione di Perl nel 1994 per la gestione del curriculum vitae dell’ideatore.
- Nel 1997 è stato rilasciato PHP 3.0 con un nuovo engine
- A partire dalla versione 4, la società Zend si è occupata di incrementarne le funzionalità aggiungendo funzionalità importanti e business-oriented (OOP, Oracle Support, PHP SPL)
- La versione attuale è la 5.6.4

L’architettura

- PHP si interfaccia al server web tramite un livello di astrazione (SAPI) che permette l’utilizzo con i più comuni web server.
- Il PHP core si occupa dei costrutti di base del linguaggio
- Lo Zend core aggiunge tutte le funzionalità avanzate (OOP e OpCode caching)
- Le estensioni sono isolate dal resto attraverso una API di estensione. Esse possono interfacciarsi con tutti e tre i livelli ed aggiungere nuove funzionalità.



Script PHP

- Gli script PHP sono file di testo la cui estensione è «**.php**»
- In combinazione con un server web, uno script PHP può includere sia codice PHP che HTML/CSS/Javascript.
- Il codice PHP è separato dal resto tramite il tag
  <?php … ?>
- Come in C e Java ogni istruzione deve terminare con il «;»
- Il server riconosce il codice PHP, lo interpreta e il risultato è restituito al browser. Il sorgente non è visibile.

Primo esempio

```
<html>
    <head>
        <title>Hello World</title>
    </head>
    <body>
        <p>
        <?php
                $myvar = "Hello from PHP!";
                echo $myvar;
        ?>
        </p>
    </body>
</html>
```

Come realizzare una pagina web

- Si possono integrare parti di codice dinamiche con il normale codice HTML:
  - <body>

    <?php $myvar = "Hello World!";

    echo $myvar; ?>

    </body>
- O si può generare tutto il codice HTML da PHP:
  - <?php

    echo "<html><head>…

    …
  - In quest'ultimo caso si può omettere il tag di chiusura «**?>**»

Parte IIb

La sintassi

Assegnazione di variabili

```php
<?php

$hello = 'Hello from PHP.';

echo $hello;
```

Commenti

```
<?php

// A one line comment

# Another one line comment

/*

A multi-line comment

*/
```

Tipi di dati primitivi

```php
<?php

$isPhpProgrammer = true; // boolean

$howOldIsPhp = 15; // integer

$pi = 3.14; // float

$event = ‘This is PHP'; // string
```

Istruzioni condizionali - If

```php
<?php
if (true) {
    echo 'Yes';
}

if (false) {
    echo 'No';
} else {
    echo 'Yes';
}
```

```php
<?php
if (false) {
    echo 'No';
} elseif (false) {
    echo 'No';
} else {
    echo 'Yes';
}
```

Istruzioni condizionali - Switch

```php
<?php
switch ($something) {
        case 'Ruby':
                echo 'No';
                break;
        case 'PHP':
                echo 'Yes';
                break;
        default:
                echo ‘I don\’t know’;
}
```

Operatori Aritmetici

```php
<?php
$a = 10;
$b = $a + 1; // 11
$c = $a - 1; // 9
$d = $a * 5; // 50
$e = $a / 2; // 5
$f = $a % 3; // 1
```

```php
echo $a++; // $a conterrà 11 e sarà stampato 10
echo $a--; // $a conterrà 10 e sarà stampato 11
echo ++$a; // $a conterrà 11 e sarà stampato 11
echo --$a; // $a conterrà 10 e sarà stampato 10
```

Concatenazione di stringhe

```php
<?php

$myString = 'foo' . 'bar'; // foobar

$myString .= 'baz'; // foobarbaz
```

Operatori di confronto

```php
<?php

if (2 == 3) { echo 'No'; }

if (3 == '3') { echo 'Yes'; }

if (2 != 3) { echo 'Yes'; }
```

```php
<?php

if (3 === 3) { echo ‘Yes'; }

if (3 === '3') { echo ‘No'; }

if (2 !== 3) { echo 'Yes'; }
```

Gli altri operatori di confronto sono identici a Java/C (>,<,<=,>=)

Operatori logici

```php
<?php

// NOT

if (!true) { echo 'No'; }

// AND

if (true && false) { echo 'No'; }

// OR

if (true || false) { echo 'No'; }
```

Le stringhe

```php
<?php
$x = 2;
echo 'I ate $x cookies.'; // I ate $x cookies.
echo "I ate $x cookies."; // I ate 2 cookies.
echo "I ate \$x cookies."; // I ate $x cookies.
echo "I ate {$x} cookies."; // I ate 2 cookies.
```

Costanti

```php
<?php

define('HELLO', 'Hello');

const HELLO = ' World';

echo HELLO . WORLD; // Hello World
```

Array Enumerativi

```php
<?php

$foo = array();

$foo[] = 'bar'; // [0] => bar

$foo[] = 'baz'; // [1] => baz


$foo = array();

$foo[0] = 'bar'; // [0] => bar

$foo[1] = 'baz'; // [1] => baz
```

```php
<?php
$foo = array(
        'bar', // [0] => bar
        'baz', // [1] => baz
);

$foo = array(
        0 => 'bar', // [0] => bar
        1 => 'baz', // [1] => baz
);
```

Array associativi

```php
<?php
$foo = array(
    'a' => 'bar', // [a] => bar
    'b' => 'baz', // [b] => baz
);
$foo['a'] = 'bar'; // [a] => bar
$foo['b'] = 'baz'; // [b] => baz
```

Cicli

```php
<?php

$x = 0;
while ($x < 5) {
        echo '.';
        $x++;
}

for ($x = 0; $x < 5; $x++) {
        echo '.';
}
```

```php
<?php
$x = array(0, 1, 2, 3, 4);
foreach ($x as $y) {
        echo $y;
}
$talks = array(
        'php'  => 'Intro to PHP',
        'ruby' => 'Intro to Ruby',
);
foreach ($talks as $id =>
$name) {
        echo "$name is talk ID
$id.\n";
}
```

Funzioni

```php
<?php
echo strlen('Hello'); // 5
echo trim(' Hello '); // Hello
echo count(array(0, 1, 2, 3)); // 4
echo uniqid(); // 4c8a6660519d5
echo mt_rand(0, 9); // 3
echo serialize(42); // i:42;
echo json_encode(array('a' => 'b')); // {"a":"b"}
function add($x, $y=3) {
    return $x + $y;
}
echo add(2, 4); // 6
echo add(4); // 7
```

Funzioni Anonime (Closures)

```php
<?php
$sayHi = function () {
        return 'Hi';
};
echo $sayHi(); // Hi

$values = array(3, 7, 2);
usort($values, function ($a, $b) {
        if ($a == $b) { return 0; }
        return ($a < $b) ? -1 : 1;
});
```

Classi e oggetti

```php
<?php
class Car {
        const ENGINE_V4 = 'V4';
        const ENGINE_V6 = 'V6';
        const ENGINE_V8 = 'V8';
        private $_hasSunroof = true;
        public function __construct($hasSunroof) {
                $this->_hasSunroof = $hasSunroof;
        }
        public function hasSunroof() {
                return $this->_hasSunroof;
        }
}
```

Classi e oggetti

```php
<?php

$myCar = new Car();

if ($myCar->hasSunroof()) {
    echo 'Yay!';
}
```

Ereditarietà e Interfacce

```php
<?php
class Chevy extends Car { }


interface Vehicle {
        public function hasSunroof();
}
class Car implements Vehicle {
        public function hasSunroof(){
                return $this->_hasSunroof;
        }
}
```

Visibilità

- «**public**» visibile ovunque (default)
- «**protected**» visibile a se stesso e ai figli e agli oggetti
- «**private**» visibile solo a se stesso

Inclusioni di altri script

- PHP permette l'inclusione di altri script PHP in un file esterno attraverso apposite funzioni:
  - **require**(), **include**(), **include_once**(), **require_once**()
- Questo permette di riusare codice in più pagine (ad esempio classi).

THE END