CIRO RAGONE

PROGRAMMARE PER IL WEB

Graphic designer: Mario Toriello

CIRO RAGONE

PROGRAMMAZIONE PER IL WEB:

Lato Client e Lato Server

Linguaggi di riferimento:
HTML, CSS, PHP + MySQL

Se intendi condividere questo ebook con un'altra persona, ti chiedo di scaricare una copia a pagamento per ciascuna delle persone a cui lo vuoi destinare. Qualora tu stia leggendo questo testo senza aver acquistato una copia, ti chiedo di acquistarne una copia, per permettermi di far crescere il lavoro e di offrire sempre più titoli con qualità elevata. Grazie per il tuo aiuto e per aver rispettato il lavoro dell'autore.

Data la frequenza degli aggiornamenti degli strumenti di sviluppo e dei

linguaggi, i contenuti si intendono fedeli allo stato dell'arte al momento della pubblicazione.
Puoi aiutare l'autore a migliorare tale guida, lasciando una recensione descrivendone i pregi e i difetti.

INDICE

HTML

7. Link
    - href
        - Inviare mail (mailto)
        - Chiamare numeri telefonici (tel)
        - download
        - Sezioni
    - Target
    - Creare Ancore
8. TABELLE
9. MODULI
    - Form
    - Input

        - color
        - date
        - datetime-local
        - month
        - week
        - time
        - number
        - email
        - search
        - range
    - Associazione Attributo - Type
    - Raggruppamento Controlli

- progress
- ruby, rt, rp

PHP

1. INTRODUZIONE
2. PROGRAMMARE IN PHP
    - Hello,World
3. VARIABILI
    - Dichiarazione
    - Tipi Di Variabili
    - Settare le variabili
    - Identificare le variabili

- Connessione al DB
- Chisura Connessione
- Esecuzione Query
- Esaminare i risultati
- All together now
- Intabellare dati a partire da qualsiasi query

1. INTRODUZIONE
    - Definizione Stile CSS
        - Stile inline
        - Stile Internal
        - Stile Esterno
        - Ereditarietà
    - Scrivere Regole CSS
2. Tipi Di Dati
    - <number>
    - <percentage>
    - <string>
    - <url>
    - <color>
    - <time>

- <length>
  - Lunghezze Relative
  - Lunghezze Assolute
- Altri tipi

3. PROPRIETA' Testo
   - Colore
   - Allineamento
   - Decorazione
   - Trasformazione del testo
   - Indentazione
   - Spaziatura tra lettere

- Spaziatura tra Parole
- Spaziatura tra Paragrafi
- Ritorno a capo
- Spezzare lunghe parole
- Ritorno a capo di parole
- Stile per spazio finito

4. PROPRIETA' Background
    - Colore
    - Immagine
    - Ripetizione Immagine
    - Dimensione Immagine

- Immagine In Movimento
- Occupazione area background

5. PROPRIETA' Font
    - Famiglia
    - Stile
    - Spessore
    - Tutto Maiuscolo
    - Dimensione
    - All In One
    - Font Personalizzato
6. PROPRIETA' Liste
    - Immagine

. . .

9. ID e CLASS
    - CLASS
    - ID
10. PSEUDOCLASSI
    - Links
    - Classi CSS e Pseudo-Classi
    - Pseudo-Classi
        - :checked
        - :disabled, :enabled
        - :valid, :invalid
        - :required,

:optional

- :in-range, :out-of-range
- :focus
- :empty
- :first-child
- :first-of-type
- :last-child e :last-of-type
- :only-child
- :only-type
- :nth-child(n)
- :nth-last-child(n)
- :nth-of-type(n)

- Selettore Contestuali
- Selettore Figlio
- Selettore Fratelli Adiacenti

13. SELETTORE ATTRIBUTI
    - Selettore[attribute]
    - Selettore[attribute=valu
    - Selettore[attribute~=va
    - Selettore[attribute|=va
    - Selettore[attribute^=va
    - Selettore[attribute$=va
    - Selettore[attribute*=va

PREFAZIONE

Il mondo dell'informatica è in continua variazione, proponendo sempre nuove tecnologie progettate per semplificare la vita dell'uomo.
Attualmente le persone utilizzano gli smartphone per chattare, chiamare, navigare in Internet, ecc..
Abbiamo tutto il mondo a portata di click, rendendo obsoleta la carta e tutti i modi di esprimersi tramite essa.
I libri in formato cartaceo sono stati

convertiti in e-book (electronic book), la calcolatrice è quella disponibile sul cellulare, in breve la tecnologia ha conquistato il cuore delle persone.
Il modo di comunicare è quindi cambiato e distinguersi tra la massa attraverso l'utilizzo di mezzi tradizionali risulta estremamente complicato, quindi in questo testo proponiamo gli strumenti che offrono la possibilità di far sapere al mondo intero che esistiamo specificando quale prodotti stiamo pubblicizzando attraverso uno stile unico, curando i

particolari e minimizzando i costi.
Il libro, per adesso, riporta l'anticipazione dell'intero testo e discute di HTML, PHP + MySQL e CSS.
In seguito affronteremo le altre tecnologie che permettono la realizzazione di un sito ovvero JavaScript, DHTML (ovvero CSS + JavaScript + DOM). Ultimo, ma non per importanza, tratteremo i CMS (in particolare Wordpress) e accenneremo alle tecniche SEO (per posizionamento dei siti).
Si è scelto di iniziare con questi

linguaggi perchè le figure professionali maggiormente ricercate in questo settore sono due (a mio avviso): un programmatore in grado di interfacciarsi con un database MySQL e un professionista in Wordpress/Joomla/Magento...
Il prezzo è ridotto rispetto all'intera opera e chi compra adesso avrà successivamente l'intero testo senza sovrapprezzo.

HTML

1. INTRODUZIONE

1.1 Storia

HTML (HyperText Markup Language – Linguaggio di Marcatura per Ipertesto) è il linguaggio utilizzato per creare documenti visibili sul web. Questi documenti testuali vengono interpretati da software chiamati browser (Internet Explorer, Firefox..) i quali mostrano le informazioni testuali in modo visuale.

L'HTML è stato sviluppato verso la fine

degli anni ottanta del XX secolo da Tim Berners-Lee al CERN di Ginevra assieme al protocollo HTTP dedicato al trasferimento di documenti in tale formato.

Il protocollo HTTP funziona su un'architettura client/server ovvero un PC (il client) effettua una richiesta ad un altro PC (il server) richiedendo qualche servizio. Il server prepara la risposta e la invia al client.

Nel nostro caso il browser (client) fa richieste ad un sito web (server).

1.2 Strumenti per Sviluppare

HTML è un linguaggio molto pratico e non richiede un elevato grado di esperienza nella programmazione. Procediamo quindi ad esaminare quali sono gli strumenti necessari per iniziare al meglio quest'avventura.

Dapprima scaricare un browser, per programmare preferisco sicuramente Mozilla Firefox poichè fornisce svariati plug-in, molti utili al programmatore per svolgere il proprio compito.

Attualmente il link per il download di questo browser il cui logo è una volpe di fuoco (Fire=fuoco, fox=volpe) ma nel nome si riferisce al panda rosso è: https://www.mozilla.org/it/firefox/des

I plug-in (componenti aggiuntive) base che ci occorrono per diventare Web Developer sono Web Developer e Firebug.

Queste componenti aggiuntive ci permetteranno di individuare, quindi risolvere, gli errori di programmazione che commetteremo.

Infine scarichiamo XAMPP, che da

quanto riporta il sito è il più popolare ambiente di sviluppo PHP.

XAMPP è una distribuzione Apache completamente gratuita facile da installare che contiene MySQL, PHP, e Perl. Il pacchetto open source XAMPP è stato concepito per un'installazione e un utilizzo intuitivi.

Infine, per scrivere il documento è possibile utilizzare degli appositi ambienti che facilitano ed assistono nella realizzazione del sito web (come ad esempio gli editor wysiwyg - what you see is what you get) oppure un

semplice documento di testo (file txt).
Preferiamo il secondo metodo perchè didattico, pratico e permette di esaminare fino in fondo le istruzioni.
Effettuiamo il download dal sito ufficiale di Notepad++ , scegliendo il pacchetto Notepad++ Installer cosi da aver più probabilità di avere successo nell'installazione.
Altro editor molto utilizzato nella comunità informatica è Sublime Text.

2. PROGRAMMARE IN HTML

2.1 Hello,World

Gli elementi fondamentali per poter scrivere un documento HTML sono pochi, in teoria nessuno. Potremmo scrivere la nostra prima pagina inserendo semplicemente del testo in un documento.

Su Windows, creiamo un "Nuovo Documento di Testo" cliccando con il

tasto destro del mouse e scegliendo dal menù Nuovo > Documento di Testo.

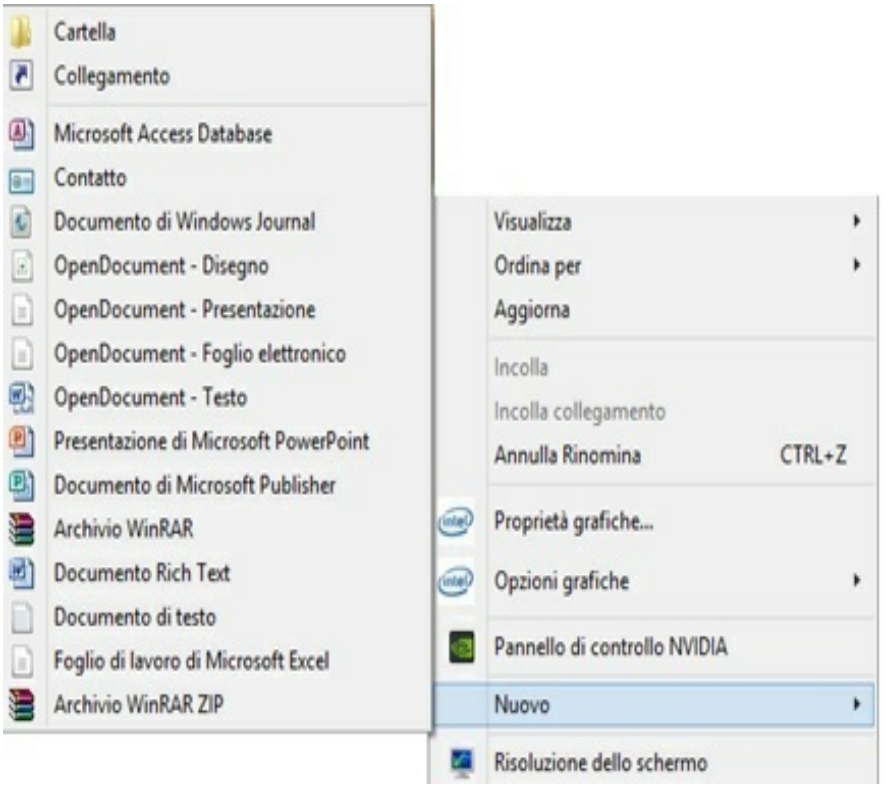


Figura 1. Creazione di un documento di testo

Apriamo quindi il documento e scriviamo una frase all'interno, come ad esempio la classica frase di saluto degli informatici "Hello,World" (per approndire clicca qui), clicchiamo File >> Salva Con Nome, e in *Salva Come* posizionato sotto*Nome File* selezioniamo *"Tutti i file (*.*)"*, inseriamo un nome (ad esempio HelloWorld), aggiungiamo *.html* e salviamo.

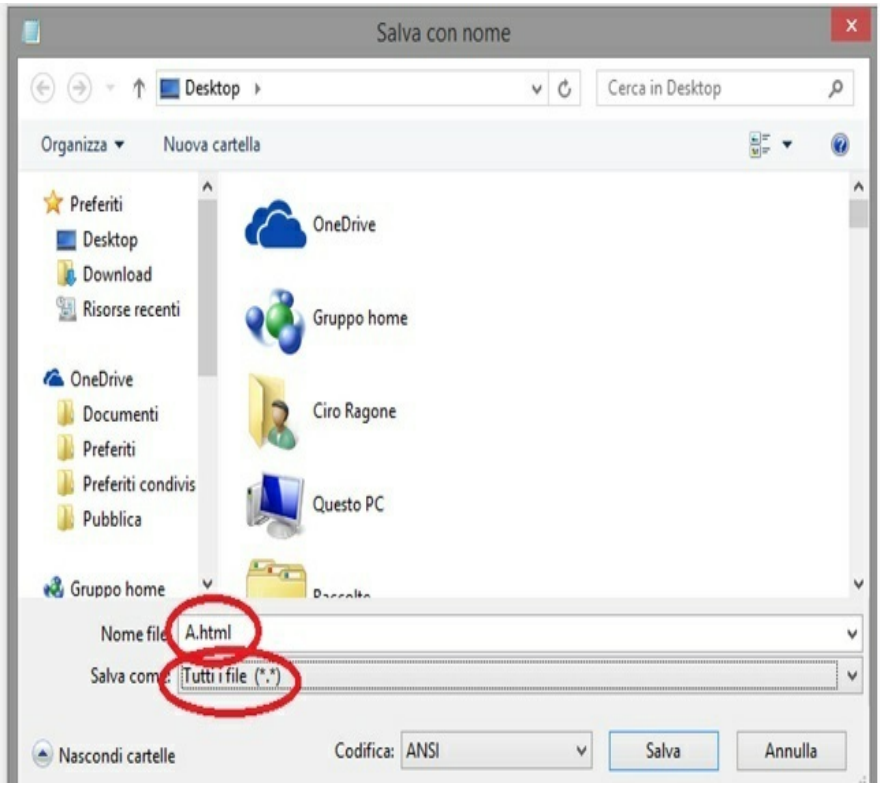


Figura 2.

Salvataggio di un file con estensione .html

Apriamo ora il nostro file con un click del mouse, notiamo che il programma di visualizzazione predefinito è il

nostro browser. Qualora non dovesse esserlo, cliccate il tasto destro del mouse sul file, selezionate apri con e scegliete il vostro browser. Ecco il nostro primo documento per il web.

2.2 Tags

Per scrivere un documento HTML è necessario utilizzare i tags. In generale, il concetto di tag è molto presente in informatica infatti è utilizzato per marcare determinati elementi o migliorare la leggibilità di

un documento da parte di un umano e/o di un calcolatore. Nel nostro caso permettono al browser di distinguere quali elementi debbano essere analizzati all'interno del documento. Sono riconoscibili poichè circondati dalle parentesi angolari (<tag> - minore e maggiore). Un tag è quindi un mezzo necessario per contrassegnare gli elementi assegnando loro determinate caratteristiche e attributi. Sono case insensitive (ovvero non fanno distinzione tra maiuscole e minuscole)

ma è preferibile scriverli in minuscolo per abituarci all'XML (altro linguaggio di programmazione per il WEB, e non solo). E' importante che ogni tag aperto venga anche chiuso, e ciò deve essere effettuato in ordine inverso all'ordine di apertura, ovvero che si chiuda dall'ultimo tag al primo, come mostriamo nell'esempio (si suppone che esistano dei tags a, b, c, d).

<a>CIAO<b><c><d>MONDO</d></

Come si può notare abbiamo chiuso prima d (l'ultimo), poi c e poi b (il primo).

E' possibile aprire e chiudere un tag contemporaneamente scrivendo <tag />.

Per aggiungere ulteriori informazioni (come ad esempio la dimensione o il colore del carattere) si usano gli attributi. Ogni tag ha diversi attributi. Analizzeremo per ogni tag gli attributi fondamentali.

2.3 Struttura Documento

Forse vi starete domando, è davvero

cosi semplice? La risposta è affermativa, poichè i navigatori correggono, laddove possibile, le informazioni sbagliate, altrimenti le ignorano e aggiungono ciò che manca per una corretta visualizzazione. Apriamo il documento HelloWorld.html con Firefox, clicchiamo con il tasto destro del mouse e dal menù selezioniamo "Analizza Elemento". Nella nuova schermata che si apre possiamo notare il seguente testo

```
<html>
<head></head>
<body>
Hello,World
</body>
</html>
```

Si può notare che sono stati aggiunti in automatico i tags mancanti.

Il primo tag che apre ufficialmente un documento è <html> . All'interno ci sono altri due tags:

- <head> : Intestazione del file.
- <body> : Corpo del programma.

2.3.1 DOCTYPE

Ogni documento html deve cominciare con <!DOCTYPE html> per indicare al browser che scriveremo il nostro documento per HTML5 cosi che possa mostrarlo in maniera adeguata. Si noti che <!DOCTYPE> non è un tag ma una dichiarazione. Ci sono altre dichiarazione ma noi utilizziamo questa.
Il passo successivo dovrebbe essere l'apertura del tag <html> specificando quale lingua stiamo

utilizzando attraverso l'attributo lang = "codice" . Per il nostro Paese utilizziamo il codice="it_IT". Scriveremo quindi <html lang = "it_IT" > .

Altro attributo interessante per il tag <html> è manifest = "nome_file" HTML5 il quale permette di specificare un file da considerare nel momento in cui l'utente vada successivamente offline. In altre parole manifest offre la possibilità di far visualizzare all'utente che ha già visitato il nostro sito di farlo anche

offline. Non mi soffermo su questo attributo perchè leggermente più avanzato però è bene conoscerne l'esistenza.

Il tag html, sebbene sia opzionale in HTML5, rappresenta la radice del documento e non dovrebbe essere omesso anche per questioni di compatibilità. Si considera il documento HTML come un albero, l'elemento padre è definito radice e tramite esso ci è permessa la navigazione dei figli e quindi di tutto l'albero.

2.3.2 Tag head

Fornisce informazioni sul documento infatti contiene il titolo del documento ed altre informazioni non visualizzate dal browser, come titolo della pagina, parole chiavi per i motori di ricerca e i file esterni da aggiungere.

- <title> : Assegnare titolo alla pagina (es.<title>Esempio Pagina Web</title>)

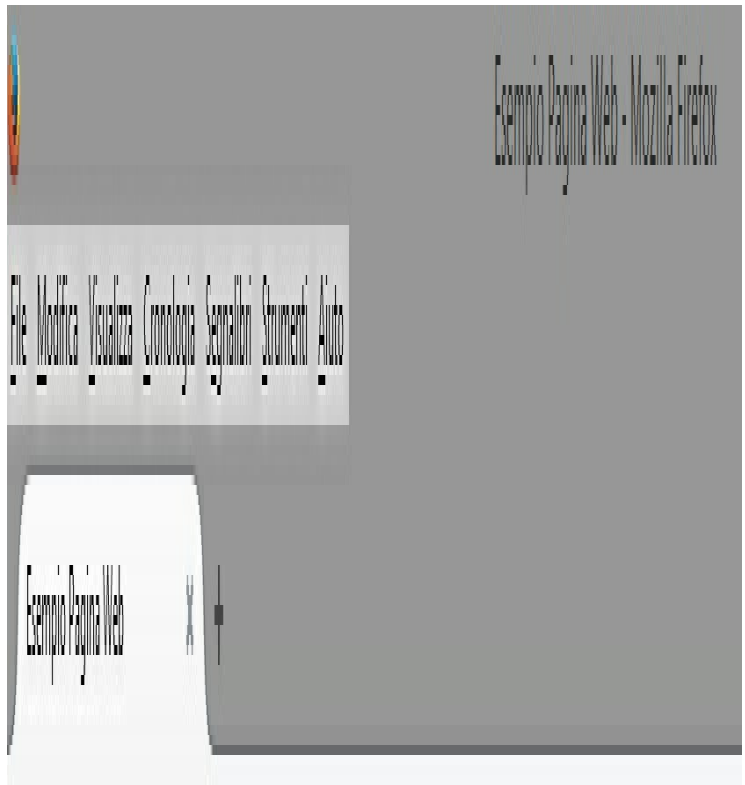


Figura 3. Aggiunta titolo alla pagina WEB

- <style> : Definire lo stile - CSS - per gli elementi della pagina (appronfondito

successivamente).

- <link> : Caricare fogli di stile esterni al documento.
- <script> : Definisce uno script client-side da importare - come ad esempio JavaScript.
- <noscript> : Definisce il contenuto da mostrare nel caso in cui lo script non sia supportato dal browser (es. JavaScript disabilitato).
- base : Poco utilizzato soprattutto dai neofiti, permette di specificare la locazione iniziale

per le URL relative (non completamente specificate con protocollo e nome dell'host)

Si usa <base href="..." />

ove href indica la stringa da prependere a tutte le URL contenute nel documento.

Esempio <base href="http://www.miosito.it/im />

indica che a tutti i link relativi viene aggiunta all'inizio questa stringa.

- <meta> : Dare informazioni riguardanti il documento. Usato

nel seguente modo:

```
<meta  name = "key"
content = "value"  />
```

I valori possibili per name e content sono:

Name	Content
Author	Autore/i documento.
Description	Sommario del documento.
Keywords	Parole chiavi per indicizzazione

Generator	Indicare con quale software è stato generato
Charset HTML5	Codifica caratteri (impostare a "UTF-8")

Es. <meta name = "author" content = "Ciro Ragone" />
<meta charset = "utf-8" /> .

E' possibile usare il tag meta anche per caricare pagine in maniera automatica utilizzando

l'attributo http-equiv .

Per ricaricare lo stesso documento dopo x secondi:

```
<meta http-equiv = "refresh"
content = "x" />
```

Per aprire in automatico una nuova pagina y dopo x secondi:

```
<meta http-equiv = "refresh"
content = "x;url=y" />
```

2.3.3 Tag body

Contiene il corpo del documento. In esso vengono inseriti i tag e tutti gli

elementi che vogliamo mostrare. Non obbligatorio in *HTML5* ma fortemente consigliato.

Diversi erano gli attributi di questo tag, soprattutto per la gestione del colore dei link e dello sfondo ma in HTML5 sono stati deprecati.

3. ELEMENTI HTML

Prima di visitare i vari tag più comuni dell'HTML, vorrei precisare che tutti i tag sono dotati di un attributo id . Tale attributo permette di associare all'elemento un identificativo univoco attraverso il quale si può successivamente assegnare degli stili (formattazione tramite CSS) oppure recuperarne il valore (tramite JavaScript).

Altro attributo generale è style con il quale è possibile definire uno stile

solo per quell'elemento (approfondiremo al momento del CSS). <tagHTML id = "nome_identificatore" >

3.1 Titoli/Intestazioni

Esistono 6 livelli di titoli: da <h1> a <h6> .

Servono per dividere il documento in "capitoli", "sezioni", "sottosezioni".

<h3> Titolo di livello 3 <h3>

Seguono esempi dei vari livelli.

Titolo di livello 1

Titolo di livello 2

Titolo di livello 3

Titolo di livello 4

Titolo di livello 5

Titolo di livello 6

Suggerimento SEO: Per Google (ed in generale i motori di ricerca) il titolo costituisce uno degli elementi con cui indicizzare la pagina, quindi ponete sempre in questo tag una frase che contenga le parole chiavi che pensate che l'utente possa

digitare quando cerca l'argomento da voi proposto.

3.2 Paragrafi

Il browser non riconosce da solo la formattazione del testo infatti ignora spazi e ritorno a capi, quindi dobbiamo indicarglieli. Se non viene indicato diversamente, esso visualizza i caratteri uno dietro l'altro anche se all'interno del nostro documento siamo andati a capo o abbiamo utilizzato più spazi.

Per indicare un paragrafo si usa il tag

<p> .

```
Esempio p.<p>NONOSTANTE SCRIVO

POSSO LASCIARE ANCHE DIVERSE R
o          DIVERSI SPAZI, MA N
</p>
```

risulta in output:

```
Esempio p.
NONOSTANTE SCRIVO SULLA STESSA
```

Per mostrare il testo esattamente come scritto nell'editor si utilizza il tag <pre>

```
Esempio pre.<pre>SCRIVENDO SUL

POSSO LASCIARE ANCHE DIVERSE R
o            DIVERSI SPAZI.
</pre>
```

risulta in output:

```
Esempio pre. SCRIVENDO SULLA S
```

POSSO LASCIARE ANCHE DIVERSE R
o DIVERSI SPAZI.

3.3 Andare a capo

Come detto è necessario indicare al browser di andare a capo. Per far ciò si usa il tag
 .

Come già detto un tag scritto come <tag/> viene aperto e chiuso contemporaneamente ed evita di

scrivere <tag> </tag>

Nel caso di parole veramente lunghe possiamo istruire il browser su come andare a capo nel momento in cui lo spazio termina, utilizzando il tag <wbr> HTML5, che non va chiuso. Va inserito in ogni punto in cui si vuole spezzare la parola.

Esempio - Spezzo la parola ogni super, cali, fragili, stiche, spiralidoso e il KINDLE spezza la linea rispettando i

miei criteri.:

```
<p>
super<wbr>cali<wbr>fragili<wbr
</p>
```

che risulta

Parola necessaria:supercalifragilistiche spiralidosupercalifragilistiche spiralidosupercalifragilistiche spiralidosupercalifragilistiche spiralidoso

3.4 Citazione

Per effettuare una citazione si usa <blockquote> e per indicare la sorgente si usa l'attributo cite .

Es. <blockquote cite = "http://www.miosito.com" > E di citazione in blockquote </blockquote>

produce:

Esempio di citazione in blockquote

Per citazioni brevi si usa invece il tag <q> , anche ad esso è associato l'attributo cite il cui utilizzo è stato appena discusso.

Es. <q

cite = "http://www.miosito.com" > E
di citazione utilizzanod q. Notate le
virgolette. </q>

produce:

"Esempio di citazione in q. Notate le
virgolette."

3.5 Documento in sezioni

Un tag ampiamente utilizzato in HTML è div .

Per separare il documento in sezioni,

per permettere di assegnare stile a diversi blocchi di una pagina si usa div . Da utilizzare nel momento in cui altri tag non costituiscono una scelta adeguata.

Esempio DIV NONOSTANTE SCR

risulta in output:

Esempio DIV
NONOSTANTE SCRIVO SULLA STESSA

3.6 Raggruppare

elementi inline

E se invece avessimo la necessità di assegnare uno stile o raggruppare solo piccole porzioni di testo?

Si utilizza il tag span . Tramite esso il normale flusso di scrittura non viene interrotto.

I vari cambi di colore in questo testo per evidenziare le parole chiavi come span, div o tutti gli altri elementi colorati di blu sono stati effettuabili proprio grazie all'utilizzo del tag span .

L'attributo principale di span è class (il quale è un **attributo generale**) attraverso il quale si esprime a quale regola CSS fare riferimento (capiremo nella III sezione del testo).

Per adesso affermo solo che

class = "token keywordHTML"

significa aver scritto da qualche parte la classe "token keywordHTML" che definisce come il testo debba essere mostrato.

Esempio span: <span class="tok

risulta essere

Esempio span: **CIAO**

3.7 Formattazione Testo

HTML ha due tipi di stile per il testo: Logico: Formatta del testo a seconda del suo significato. Lascia al browser l'interpretazione del comando secondo le proprie capacità. In altre parole, riguarda un aspetto puramente grafico.

Fisico: Indica al browser il formato specifico del testo forzandolo ad avere un particolare aspetto cosi da marcare determinati elementi.

3.7.1 Tags Logici

TAG	Come Appare
em - enfasi	*Esempio di testo*
strong - forte enfasi	**Esempio di testo**
address - per indirizzi	*Esempio di testo*

blockquote - per citazione	Esempio di testo
code - per codice	`var somma = 1 + 1;`

Altro tag logico è abbr , che il KINDLE non mostra correttamente quindi allego un immagine. Tale tag viene utilizzato per esplicare abbreviazioni ed acronimi specificando il significato attraverso l'attributo title .

Es. <abbr title = "HyperText

Markup
Language" > HTML </abbr>

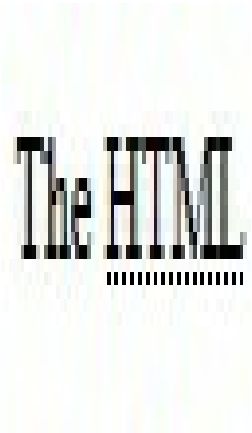


Figura 4. Esempio del tag abbr

Si noti che title è un **attributo generale** quindi qualsiasi tag può

essere accompagnato da una descrizione. Nel momento in cui l'utente passa sopra il testo racchiusto dal tag col title settato, il testo aggiuntivo viene mostrato.

3.7.2 Tags Fisici

TAG	Come Appare
b - grassetto	**Esempio di testo**
i - italico	*Esempio di testo*

u - sottolineato	<u>Esempio di testo</u>
s - barrato	~~Esempio di testo~~
small - testo più piccolo	Esempio di testo
mark	Esempio di <mark>testo</mark>
sub - pedice	Esempio di _{testo}
sup - apice	Esempio di ^{testo}

E' facile fare confusione nell'utilizzo dei diversi tags. Proviamo a fare un po di chiarezza.

Conviene utilizzare strong per testo di una certa importanza, em per porre emfasi, mark HTML5 per evidenziare il testo. Per i titoli e le intestazioni, usare h1..h6. Nel momento in cui nessuna di questa condizione è applicabile utilizzare b.

Altro elemento di confusione potrebbe essere la distinzione tra i ed em poichè vengono mostrati dal browser allo stesso modo. Il primo si

utilizza per del testo che si differenzia dal normale testo, come ad esempio il nome di un titolo o di un film, mentre il secondo per dare enfasi. Per fare un esempio concreto, si utilizza i nel seguente caso: *Il Padrino* è una bellissima trilogia, mentre uso em nel seguente caso: *Spara,* non preoccuparti delle conseguenze.
In questo caso sto ponendo una marcatura alla parola "Spara".

3.8 Scrivere codice in

HTML

Per scrivere codice in HTML, è possibile utilizzare il tag <code> in combinazione con:

- <kdb> : per definire input proveniente dall'utente.
- <samp> : per definire l'output proveniente da un programma.
- <var> : per definire una variabile.
- <pre> : per preservare linee, spazi, ritorno a capo.

```
<pre><code>Pseudocodice per Le

<var>int N;</var>
<kbd>Leggi N</kbd>
<samp>Stampa N</var></pre>
```

che risulta essere:

```
Pseudocodice per Leggere da tast:
```

int N;

```
Leggi N
Stampa N
```

3.9 Commentare in

HTML

Per commentare in HTML, è necessario circondare il testo con questi simboli:<!-- -->

Es. <!-- Testo che voglio commentare -->

produce in output:

ovvero un bel niente essendo il testo e i tag commentati.

I commenti si usano per tralasciare momentaneamente degli elementi affinchè questi non vengano mostrati

oppure per scrivere documentazione, ovvero indicare in quelle linee di codice qual è il nostro intento.

4. LISTE

E' possibile avere diversi tipi di elenchi: puntati, numerati, annidati, menu, glossario

4.1 Elenchi Puntati

Per indicare elenchi puntati (unordered list) si usa <ul>... e ciascun elemento dell'elenco (list item) deve essere compreso tra i tag <li> Descrizione <li> .

Esempio:

```
<ul>
<li>Primo Elemento<li>
<li>Secondo Elemento<li>
<li>Terzo Elemento<li>
</ul>
```

produce in output

- **Primo Elemento**
- **Secondo Elemento**
- **Terzo Elemento**

4.2 Elenchi Numerati

Per indicare elenchi numerati

(ordered list) si usa <ol>.. e ciascun elemento dell'elenco deve essere compreso tra i tag <li> Descrizione <li> .

Se si vuole definire il valore iniziale da cui partire si usa l'attributo value = "numero" da inserire nel primo

Esempio:

```
<ol>
<li value="3">Primo Elemento<li>
<li>Secondo Elemento<li>
<li>Terzo Elemento<li>
```

```
</ol>
```

produce in output

3. **Primo Elemento**
4. **Secondo Elemento**
5. **Terzo Elemento**

Stesso risultato può essere ottenuto attraverso il nuovo attributo di <ol> start = "numero" HTML5 .

Esempio:

```
<ol start="3">
<li>Primo Elemento<li>
```

```
<li>Secondo Elemento<li>
<li>Terzo Elemento<li>
</ol>
```

produce in output

3. **Primo Elemento**
4. **Secondo Elemento**
5. **Terzo Elemento**

Qualora si voglia dare ordine inverso alla lista, si può utilizzare l'attributo di `<ol>` reversed HTML5 (non supportato in IE).

Esempio:

```
<ol  reversed>
<li>Primo Elemento<li>
<li>Secondo Elemento<li>
<li>Terzo Elemento<li>
</ol>
```

produce in output

3. **Primo Elemento**
2. **Secondo Elemento**
1. **Terzo Elemento**

4.3 Elenchi Annidati

Gli elenchi annidati sono elenchi di

elenchi. Elenchi non ordinati possono essere annidati in ordinati e viceversa.

Per definire il valore iniziale da cui partire si usa l'attributo

value = "numero" per il primo <li> (come già detto).

Esempio:

```
<ul>
<li>Primo Elemento<li>
<li>Secondo Elemento<li>
<ol>
<li value="3">Terzo Elemento<li>
<li>Quarto Elemento<li>
```

```
</ol></ul>
```

produce in output

- **Primo Elemento**
- **Secondo Elemento**

3. **Terzo Elemento**
4. **Quarto Elemento**

4.4 Glossario

Permettono di includere una

descrizione di ogni termine (item) della lista.

Si usa <dl>..</dl> (definition list) per aprire e chiudere la lista, <dt>..</dt> (definition term) per ogni termine da definire e per fornire la spiegazione si usa <dd>..</dd> (definition data)

Esempio:

```
<dl>
<dt> elenco numerato </dt>
<dd> gli elementi sono numerat
</dl>
```

produce in output

elenco numerato

gli elementi sono numerati in sequenz

elenco puntato

gli elementi sono puntati

5. Caratteri Speciali

In HTML alcuni caratteri hanno un senso particolare e vengono riservati ai browsers per interpretare il testo in maniera corretta.
Ad esempio, scrivere <tag> contraddistigue un tag, ma come si può notare quei simboli che circondano il tag non sono null'altro che un minore ed un maggiore.
Come facciamo quindi a scrivere minore, maggiore o altri simboli particolari?

Per risolvere questo problema facciamo riferimento ai **caratteri speciali dell'HTML**, ce ne sono diversi come indicato dal w3, ma nella tabella che segue vengono indicati quelli più comuni.

Si noti che ogni codice *deve iniziare con & e terminare con ;*.

Per scrivere le vocali accentate con accento grave: & + lettera + grave + ;

Simbolo	**Codice**
à	`&agrave;`
À	`&Agrave;`

è	è
È	È
ì	ì
Ì	Ì
ò	ò
Ò	Ò
ù	ù
Ù	Ù

Analogo discorso seguono le vocali accentate con accento acuto (acute),

circonflesso (circ), dieresi (uml)

Es: á rappresenta á

Es: â rappresenta â

Es: ä rappresenta ä.

Ho preferito indicare solo il tipo di accento (grave, acuto, ecc..) perchè in questo modo dovrebbe essere più facile da ricordare.

Altri simboli utili sono:

Simbolo	**Codice**
Spazio (Blank space)	
>	>
<	<

–	`&ndash;`
&	`&`

Note: Se volete mostrare il codice cosi come è, senza che il browser lo interpreti, come ho fatto io nella colonna di destra, molti indicano di utilizzare i tag <pre> e <code>, ma devo essere onesto (almeno a me) non sempre funzionano.
Semplice trucchetto per ottenere il risultato sperato è quello di scrivere all'interno del codice `&` e il codice

in questione.

Es: Per mostrare il codice del maggiore scrivere: >

6. Immagini

E' possibile inserire delle immagini all'interno delle pagine HTML.

Il tag necessario è <img> che richiede come attributo src = path .

Come base del path si fa riferimento alla cartella contenente il documento HTML.

L'altro tag che suggerisco è alt = "descrizione" che sta per alternative. Tale tag contiene del testo da mostrare qualora l'immmagine non fosse disponibile.

Gli altri tags interessanti sono

height = "misura" e

width = "misura" per indicare altezza e grandezza dell'immagine.

Suggerimento SEO: evitare di usare questi due tags, perchè le immagini scalate in questo modo rallentano il caricamento della pagina Web venendo declassate dagli algoritmi di ricerca come Google. E' preferibile avere le immagine già della dimensione che si intende mostrare.

Esempio: Supponendo di avere la

pagina HTML nel percorso C:\wamp\www\esempi.html e l'immagine in C:\wamp\www\img\imgTest.png è necessario scrivere:

```
<img  src = "img/imgTest.png"
alt = "Immagine Test"  />
```

Note: Per navigare all'interno della directory e risalire alla cartella superiore utilizzare .. (punto punto - senza inserire spazi).
Ad esempio se il nostro file si trova in C:\wamp\www\esempi, ma la cartella dell'immagine si trova in

C:\wamp\www\img e il suo nome è "immagine.png" allora quello che scriveremo è

src = "../img/immagine.png" ,

ovvero risalgo nella directory superiore (accedo a C:\wamp\www), mi sposto nella cartella img e quindi faccio riferimento all'immagine.

7. Link

Navigando nelle pagine web è possibile rimandare ad altre pagine attraverso dei collegamenti (link) oppure a dei documenti.

Il tag necessario per creare un'ancora è `<a>` . Il testo compreso tra il tag viene visualizzato in maniera sottolineata e risulta cliccabile, inoltre passando su tale testo (concetto di hover - sorvolare) il cursore diventa una manina.

Lo stato in cui si trova (non visitato,

attivo, visitato, ecc.) influisce sul colore del testo del link. Il colore standard per link non visitati è blu, visitato è viola, attivo è rosso. E' possibile personalizzare i colori ma lo tratteremo successivamente (con CSS).

7.1 href

L'attributo fondamentale per stabilire il percorso della pagina da aprire è

href = path .

Es: richiamare un'altra pagina web:

```
<a  href="https://www.amazon.it/PRO
```

che risulta essere IL MIO LIBRO.

Per rendere un'immagine cliccabile il trucco è semplice poichè basta circondare <img> con <a> .

Es: richiamare un'altra pagina web cliccando un'immagine:

```
<a  href="https://www.amazon.it/PRO
<img  src="copertina.png"  alt="Coper
```

che risulta essere

Come si può notare, il path dell' href può essere assoluto o relativo. Può riferirsi a risorse esterne oppure locali (presenti sul disco locale) - come visto per l'esempio dell'immagine.

Per le risorse locali l'indirizzo base che viene preposto al path indicato, è automaticamente assegnato ed è

uguale al path della cartella in cui si trova il file. Ad esempio, se il nostro file (quello in cui stiamo scrivendo il codice) si trova in C:\wamp\www\esempi e l'immagine si trova nella cartella img e il suo nome è "immagine.png" allora nell'attributo href scrivendo "img/immagine.png", automaticamente verrà interpretato come "C:\wamp\www\esempi\img\immagine

7.1.1 mailto

Per inserire un link che permetta di

inviare una email si utilizza href = "mailto:email" . Per includere più indirizzi email contemporaneamente separare ogni voce con una virgola.

```
<a  href="mailto:ragoneciro@hotmail.
```

che risulta essere Inviami una mail.

Per inserire l'oggetto della mail si utilizza "mailto:email?subject" .

```
<a  href="mailto:ragoneciro@hotmail.
```

che risulta essere Inviami una mail.

Nella seguente tabella indichiamo

tutte le possibili opzioni:

Voce	Descrizione
? subject=descrizione	Per specificare l'oggetto della mail
?body=descrizione	Per specificare il messaggio della mail
?cc=descrizione	Per specificare altri indirizzi in CC (copia carbone)
	Per specificare

?bcc=descrizione	altri indirizzi in CC nascosta

Per inserire più opzioni contemporaneamente utilizzare il simbolo & invece del ? dalla seconda voce in poi.
Volendo fare un esempio omni-comprensivo:

```
<a  href="mailto:ragoneciro@hotmail.
```

che risulta essere Inviami una mail.

7.1.2 tel

Nel momento in cui l'utente visita la nostra pagina web attraverso uno smartphone, è possibile indicare un link che se cliccato fa comparire direttamente il numero sul tastierino cosi da poter far partire la chiamata.

Per **avviare una chiamata** si utilizza href = "tel:numero" .

```
<a  href="tel:+39000000000">Chiaman
```

che risulta essere Chiamami.

7.1.3 download

E' possibile permettere il download del file, anzichè la navigazione dello stesso. Per far ciò si utilizza l'attributo

download = "nome_file" HTML5

(non supportato da Safari), dove il "nome_file" è il nome con cui il file verrà salvato sul PC dell'utente.

```
<a  href="images/pallina.png"  downlo
```

che risulta essere Download immagine pallina. (ovviamente tale opzione sul KINDLE non è attiva).

7.2 Target

Interessante è target = value il quale specifica dove aprire il documento. I possibili valori sono:

Value	Operazione
_blank	Aprire il collegamento in una nuova finestra.
_self	Aprire il collegamento nella stessa finestra, rimpiazzando il contenuto corrente.
	Aprire il collegamento nel

_parent	frame genitore
_top	Aprire il collegamento nella finestra a schermo intero.

Es: richiamare un'altra pagina web:

```
<a  href="http://www.amazon.it/PRO
```

che risulta essere IL MIO LIBRO.

E' possibile anche specificare il path che punti ad un documento del disco locale per permetterne il download.

7.3 Creare Ancore All'interno del documeno

E' possibile creare delle ancore all'interno del documento, ovvero elementi che, se cliccati, fanno navigare l'utente verso un'altra sezione.
Volendo fare un esempio concreto, l'indice che trovate all'inizio del libro è stato strutturato con questa tecnica.
Per ottenere ciò abbiamo bisogno di

due parti, il link che l'utente deve cliccare e che punta ad una certa sezione indicata tramite l'attributo name e l'altra che indica effettivamente la sezione di interesse: Quando ho dichiarato l'indice ho scritto:

```
<a name="Indice"> Questo è l'in
```

mentre nella pagina corrente inserisco il link che rimanda all'indice:

```
<a href="#Indice"> Vai all'indice
```

che risulta essere:

Vai all'indice

8. TABELLE

Il tag necessario per costruire una tabella in HTML è <table> . Una tabella si divide in 3 parti: intestazione, corpo e foot rispettivamente indicati con <thead> , <tbody> e <tfoot> .Questi tre tags sono opzionali in quanto aggiunti automaticamente dal browser.

Ogni tabella è composta da righe e colonne.

Per iniziare una nuova riga è

necessario usare <tr> .
Di solito, la prima riga di una tabella contiene l'intestazione, indicata col tag <thead> a cui segue l'apertura di una nuova riga con <tr> e ogni voce viene rappresentata tramite <th> . Terminata l'intestazione si scrive </thead>.
Se si vuole inserire un footer, questo deve essere scritto dopo il <thead> e prima del contenuto della tabelle.
Per segnalare l'inizio del footer scriviamo <tfoot> , diamo inizio ad una nuova riga con <tr> e

rappresentiamo ogni voce di questa sezione attraverso <td> . Il tag va ovviamente chiuso.

A questo punto è possibile scrivere il corpo della tabella, il cui inizio è indicato con <tbody> , come detto ogni riga della tabella è indicata dal <tr> , mentre ogni elemento viene circondato dal tag <td> . Terminato il corpo si chiude il tag <tbody> .

Come si è potuto notare i tag per rappresentare le celle sono due:

- tr : Per intestazioni - i valori sono

centrati ed in grassetto (di default).

- td : Per tutto il resto - i valori sono allineati a sinistra e scritto in maniera regolare.

Per dare un titolo alla tabella si usa il tag <caption> da inserire immediamente dopo al tag <table> . Iniziamo a costruire una tabella base:

```
<table>
<caption>Caption</caption>
<thead><tr><th>H1</th><th>H2</th><
<tfoot>
```

```
<tr><td>Foot1</td><td>Foot2</td><
</tfoot>
<tbody>
<tr><td>Val1</td><td>Val2</td></tr
</tbody>
</table>
```

che produce in output

Caption

H1	H2
Val1	Val2
Foot1	Foot2

Come si può notare questa tabella non è incorniciata ma vedremo successivamente tramite CSS come stilizzarla.

Attributi interessanti di <td> sono rowspan = "numero" e colspan = "numero" che permettono ad una cella di espandersi per il valore indicato su più righe e colonne rispettivamente.

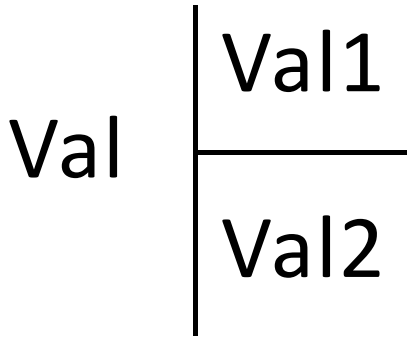
Es. tabella con uso rowspan e colspan:

```
<table border="1">
<thead><tr><th>H1</th><th>H2</th><
```

```
<tbody>
<tr><td>Val1</td><td>Val2</td></tr
<tr><td  colspan="2" >Col1 e 2</td
<tr><td  rowspan="2">Val</td>
<td>Val1</td></tr>
<tr><td>Val2</td></tr>
</tbody>
</table>
```

che produce in output

H1	H2
Val1	Val2
Col1 e 2	



Ho usato momentamente anche l'attributo border per poter mostrare l'effetto di rowspan e colspan ma in HTML5 è deprecato e sostituito dal CSS.

Note: Disponibili online ci sono molti siti web che permettono la generazione grafica della tabella e restituiscono il codice corrispondente.

9. MODULI

Premessa: Il KINDLE non interpreta i comandi input, form ecc.... Farò del mio meglio per spiegarvi ciò che accade con ogni attributo, ma vedere con i propri occhi è meglio di immaginare, per questo per un esempio live vi invito a collegarvi al link http://ciroragone.altervista.org/book/k input/ per vedere i vari comandi in azione.

9.1 Form

Finalmente analizziamo il modo attraverso il quale l'utente può interagire con una pagina web.
Un modulo è un'area della pagina predisposta per accettare dei dati in ingresso dall'utente.
I dati vengono inviati al WEB server che restituisce una risposta (pagina HTML) dopo averli elaborati.
Ogni modulo è definito dal tag <form> , seguito da tutti gli elementi (controlli) che lo costituiscono.

Il tag form quindi apre il modulo, ha diversi attributi, alcuni molto importanti.

Il primo che esaminiamo è action il quale specifica il path del file che deve processare i dati in input.

Il secondo è method che indica in che modo inviare i dati, se tramite GET o POST. Quest'ultimo è da preferire per dati sensibili in quanto i dati inviati tramite il primo metodo verranno mostrati in chiaro (nella url) e quindi visibili a tutti.

Altri due attributi da analizzare sono

name per indicare il nome del form e target (i cui valori sono già stati esaminati) per definire dove mostrare la risposta.

In HTML5 per il tag <form> sono stati aggiunti autocomplete HTML5 con valore "on","off" per indicare se attivare o meno l'autocompletamento - quando inviate dei dati a qualche sito e dopo un pò ritornate su tale sito, cliccando di nuovo sulle varie caselle per inserire le vostre informazioni, notate che esse sono già disponibili, questo è l' autocomplete

HTML5 .

Anche se autocomplete HTML5 è attivo per l'intero modulo è possibile disattivare l'autocompletamento per qualche singolo elemento scrivendo all'interno del controllo

autocomplete = "off" .

Altro nuovo attributo è il

novalidate = "novalidate" HTML5

(non supportato in Safari) per indicare di non validare il form (capiremo quando approfondiremo, ad esempio con la spiegazione di <input

type = "email" > .

9.2 Input

Input è senza dubbio uno dei tag principali. Permette di definire il campo dove l'utente inserisce i dati e, tra le altre cose, di che tipo deve essere (testo, password, ...).
Per questo motivo ne analizziamo la quasi totalità degli attributi ad esso associabili.
L'attributo sempre necessario affinchè il server possa prelevare i dati inseriti dall'utente è name = "valore" (negli esempi non sarà inserito name per

non comprottere la leggibilità).

L'attributo principale è type il quale può assumere diversi valori, ma per adesso interessiamoci unicamente a "text" che permette di utilizzare la casella per l'inserimento di testo.

Es. Esempio Text: <input type = "text" />

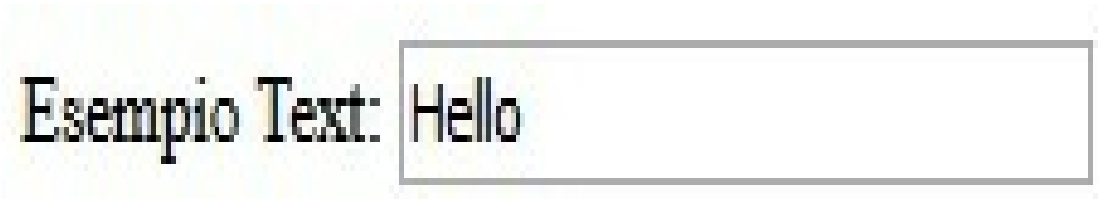
Se vogliamo, che al caricamento della pagina un qualche valore sia

già presente all'interno si valorizza l'attributo

value = "valore_iniziale" />

Es. <input type = "text"

value = "Hello" />



Se vogliamo mostrare un suggerimento all'interno della casella, che scompare nel momento in cui l'utente clicca nella casella, si usa placeholder = "value" HTML5

.

Es. <input type = "text"
placeholder = "Esempio
Placeholder" />

Se vogliamo limitare il numero di caratteri che si possono immettere c'è maxlength

Es. <input type = "text"
maxlength = "16" />

Se vogliamo che il cursore sia presente in questa casella al caricamento della pagina (come accade con Google che ha il cursore direttamente attivo nella casella di ricerca): autofocus = "autofocus"

HTML5

Es. <input type = "text" autofocus = "autofocus" />

Casella in sola lettura: readonly = "readonly" per permettere di mostrare ma non

modificare valori. Il suo valore sarà comunque passato al server quando inviato tramite GET o POST.

Es. `<input type = "text" readonly = "readonly" />`

Casella disabilitata: `disabled = "disabled"` rende non cliccabile e non utilizzabile la casella. Il suo valore se disabled non sarà passato al server .

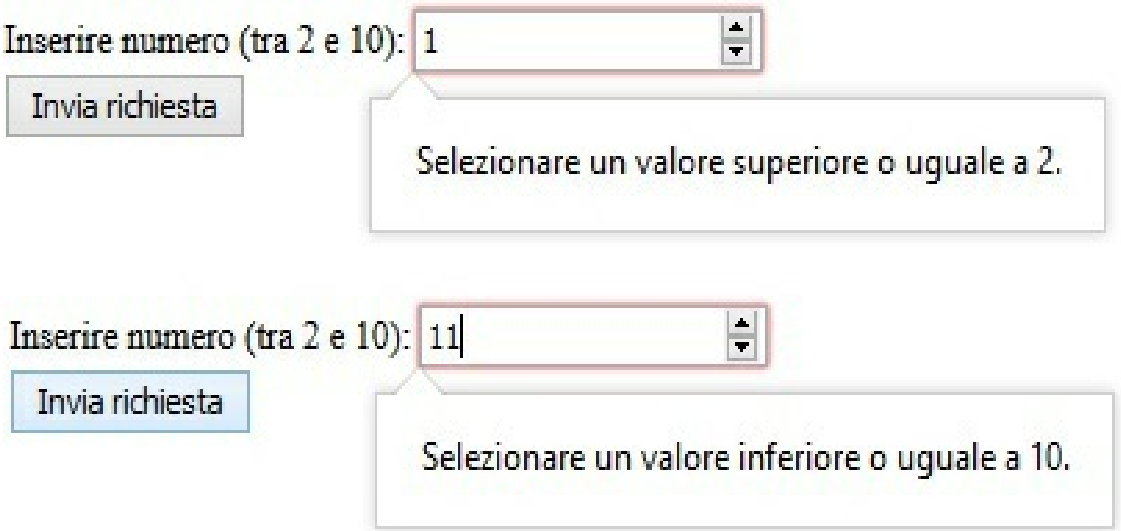
Es. `<input type = "text"`

disabled = "disabled" />

Range di valori: max HTML5 e min HTML5 permettono di definire il valore minimo e massimo utilizzabile. Non devono essere usati necessariamente entrambi, quindi si può anche scegliere di definire solo un upper o un lower bound (un limite massimo o minimo). Alla sottomissione del

modulo, i valori inseriti verrano controllati.

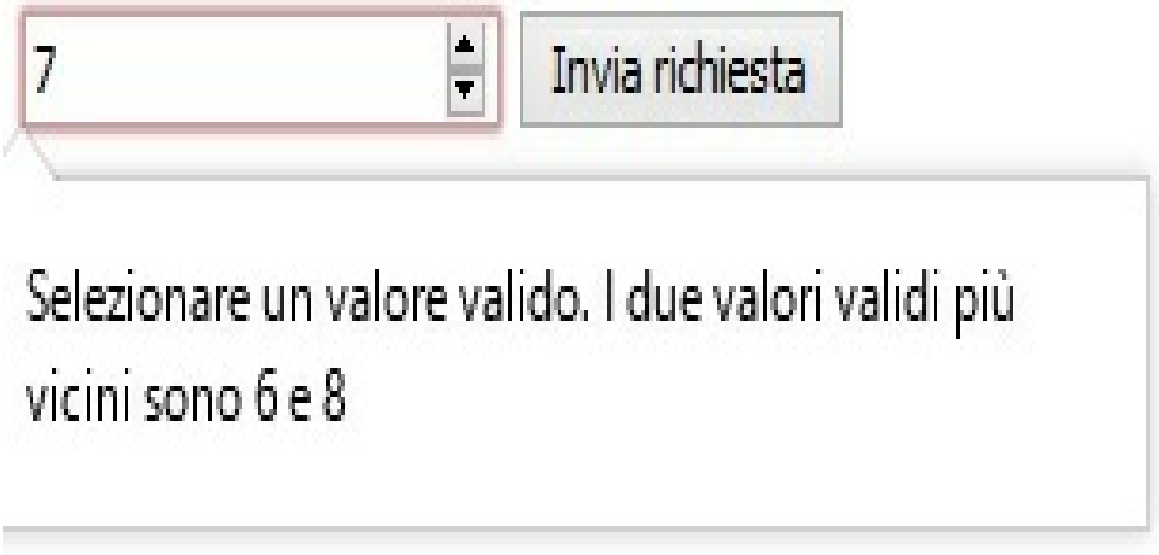
```
<input  type = "text"  max = "10"
min = "2" />
```



Passo: step HTML5 permette di accettare solo i multipli e sottomultipli del valore passato

```
<input  type = "text"  step = "2"
```

/> accetta ..,-2,0,2,..



9.3 Altri Input Type Testuali

Abbiamo analizzato <input type = "text" /> che ci permette di inserire del testo e gli attributi che ci permettono di aggiungere

informazioni, limitare valori ecc... i quali sono (quasi tutti) validi anche per gli altri input type.

Nel caso in cui volessimo permettere l'inserimento di caratteri mostrati come asterischi si usa <input type = "password" /> .

Se vogliamo salvare un valore in

una casella senza mostrarla all'utente allora type è impostato a "hidden" . Utile quando si vogliono salvare dei valori da passare successivamente al server, senza che questi siano visibili all'utente.

NOTA: Analizzando il codice HTML il valore però viene mostrato, quindi evitare di salvare all'interno dati sensibili.

Per permettere l'eliminazione di tutti i valori immessi all'interno del

form si utilizza <input type = "reset" /> . Per customizzare il messaggio mostrato sul bottone, che di default è Reset, valorizzare value .

Es. <input type = "Reset" value = "Mio Messaggio" />

9.4 Input per Selezione

Supponiamo di dover porre l'utente di fronte una scelta. Gli mostriamo diverse possibilità permettendo la

scelta di una o più voci.

9.4.1 Selezione Singola

Se le scelte da mostrare all'utente sono poche, permettendo la selezione di una sola voce, è possibile utilizzare i radio button.

Per far si che una sola voce sia scelta, è necessario che tutti i radio button coinvolti abbiano lo stesso attributo name .

Se vogliamo che una tra le voce sia impostata di default allora si utilizzi l'attributo checked = "checked" .

Ogni radio ha associato un suo "value" (univoco) cosi da poter identificare la scelta dell'utente.
Il valore di "value" e della descrizione può essere diversa, ovvero non è necessario che essi coincidano (come mostrato nell'esempio). Ciò permette di assegnare al value un valore arbitrario.
Es. Scegli come pagare:

```
<input type="radio" name="pagatoC
<input type="radio" name="pagatoC
<input type="radio" name="pagatoC
```

In questo modo una delle 3 voci è cliccabile. E' ovviamente possibile avere più gruppi di radio button, basta assegnare loro nomi differenti!

9.4.2 Selezione Multipla

Se le scelte da mostrare all'utente sono poche, permettendo la selezione di più voci, è possibile utilizzare le

checkbox.

Per far si che le voci selezionate siano correttamente elaborate, è necessario che tutte le checkbox coinvolte abbiano lo stesso attributo name concluso da [] (poichè è un vettore, vediamo poi in PHP).

Se vogliamo che una tra le voci sia impostata di default allora si utilizzi l'attributo checked = "checked" .

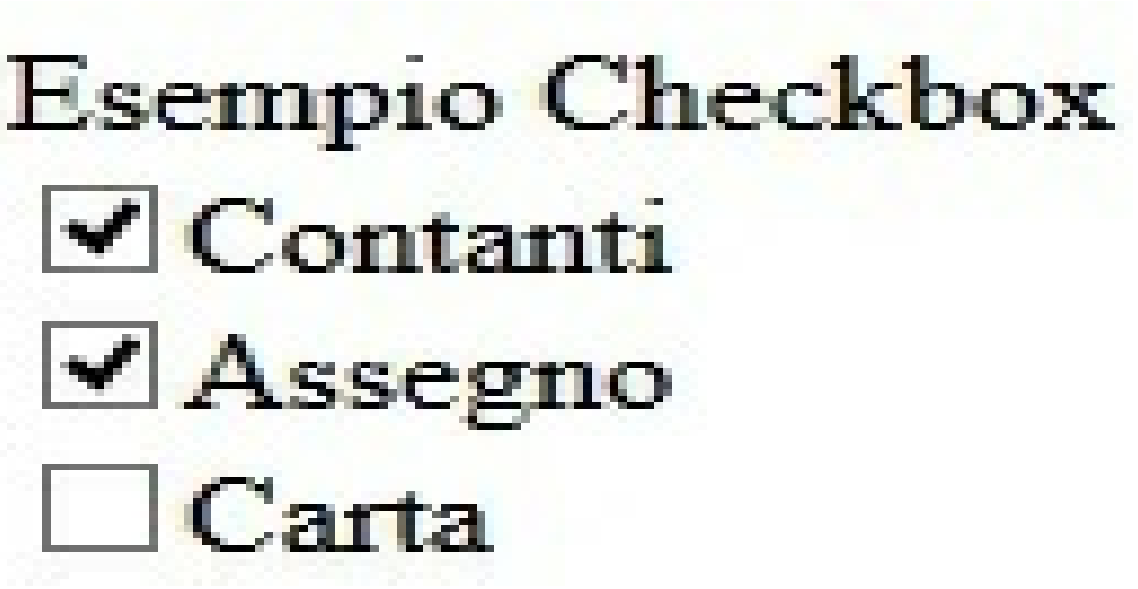
Ogni checkbox ha associato un suo value (univoco) cosi da poter identificare la scelta dell'utente.

La scelta del value e della descrizione

può essere diversa, ovvero non è necessario che essi coincidano (come mostrato nell'esempio). Ciò permette di assegnare al value un valore arbitrario.

Es. Scegli come pagare:

```
<input type="checkbox" name="paga
<input type="checkbox" name="paga
<input type="checkbox" name="paga
```



9.4.3 Selezione Con Molte Voci

Se le scelte da mostrare all'utente sono molte, sia che si voglia permettere la selezione di una voce, sia che si voglia permette la selezione multipla, è possibile utilizzare la <select> .

Il tag <select> definisce una lista di selezione. Gli attributi principali sono name (per manipolare l'elemento tramite JavaScript), size (che mostra quante voci alla volta mostrare) e multiple .

Nel momento in cui quest'ultimo attributo è presente la select funziona come checkbox (possibilità di scegliere più voci), altrimenti si lavora come radio button (selezione singola).

Per inserire valori all'interno della <select> è necessario inserire una <option> per ogni voce che

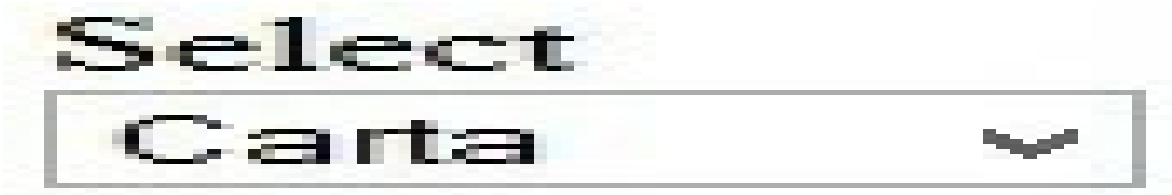
compone la selezione (come mostrato nell'esempio).

Per poter poi passare il valore al server (e quindi capire cosa sia stato scelto dall'utente) valorizzare l'attributo value. Per assegnare un valore di default alla <select> inserire nella <option> selected.

Es. Scegli come pagare:

```
<select size="3" name="pagatoCon"
<option value="1"> Contanti </op
<option value="2"> Assegno </opti
<option value="3" selected > Carta
```

```
<option value="4"> Paypal </optio
</select>
```



In questo modo una sola delle voci è cliccabile. Per cliccare più voci contemporaneamente usare l'attributo multiple e al momento della selezione tenere premuto il tasto ctrl(su windows).

Se si vuole che al caricamento della pagina il focus sia sulla <select> , si può utilizzare autofocus HTML5 (non supportato in Firefox), per disabilitare l'elemento si utilizza disabled , per forzare l'utente ad effettuare una scelta usare required HTML5 (non supportato in Safari). E' possibile anche inserire un'etichetta non cliccabile per un gruppo di voci usando <optgroup> nel seguente modo:

```
<select>
<optgroup label="Auto Svedesi">
<option value="volvo"> Volvo </op
<option value="saab"> Saab </optio
</optgroup>
<optgroup label="Auto Tedesche">
<option value="mercedes"> Mercede
<option value="audi"> Audi </optio
</optgroup>
</select>
```

9.5 Input per Upload File

Per caricare un file si usa <input type = "file" /> , che accetta come attributi multiple = "multiple" per effettuare l'upload di più file e accept = "valore" dove "valore" può essere uno dei seguenti:

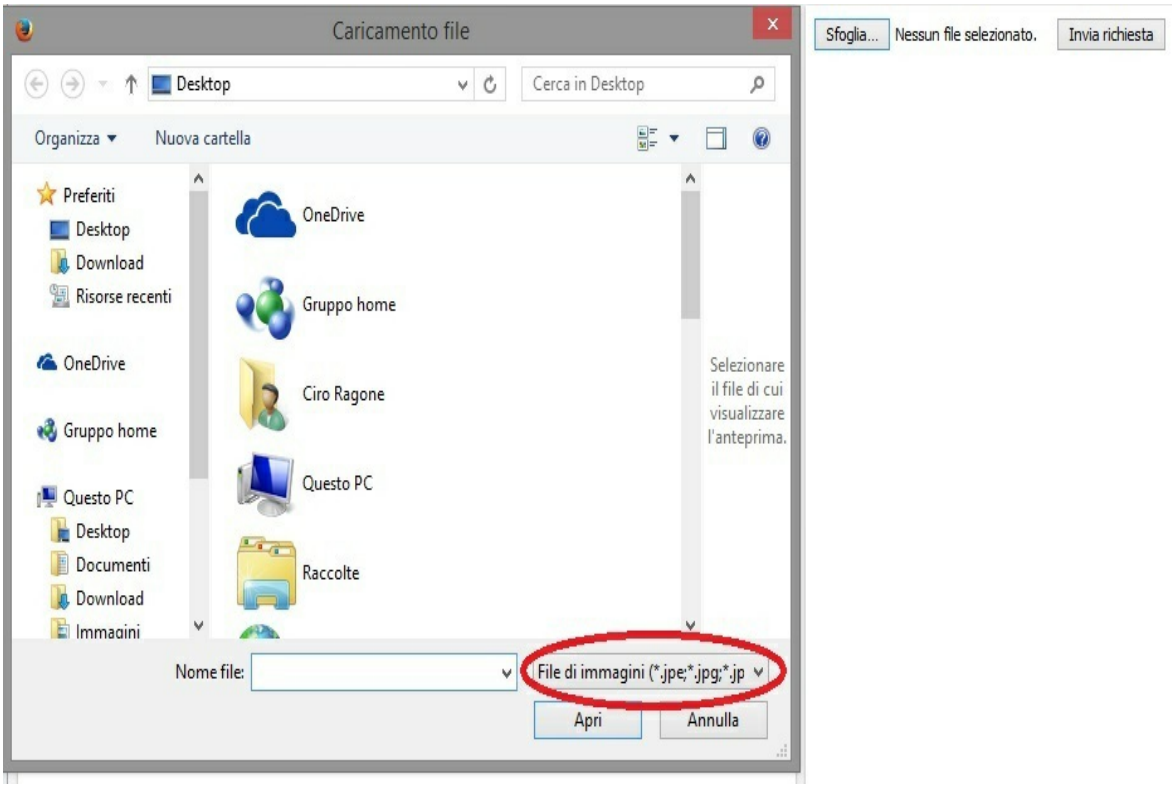
Valore
estensione file

.pdf
text/html
text/plain
application/vnd.ms-excel
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
audio/*

video/*
image/*

Permette di aprire una finestra e selezionare solo file immagini.

```
<input type="file" accept="image/*"
```



9.6 Input per Esecuzione

Vediamo ora come eseguire azioni o funzioni.

9.6.1 Eseguire funzione

Vogliamo che al click di un bottone una funzione sia eseguita. Per far ciò si usa <input type = "button" indicando a quale evento l'azione deve essere richiamato (onclick, onmouseover,...). Vedremo con JavaScript il suo significato.

9.6.2 Eseguire azione

Quando tutti i dati sono stati inseriti, l'utente può sottomettere il suo inserimento affinchè i dati vengano processati.

Per fa ciò si usa <input

type = "submit" > all'interno del form nella quale abbiamo specificato quale script (e quindi quale azione) eseguire. Per il messaggio personalizzato si usi value = "Messaggio Personalizzato" . Se invece vogliamo un'immagine a rappresentare il nostro bottone per effettuare il "submit" , possiamo utilizzare <input type = "image" > HTML5 , combinato con src = "path_immagine" e a cui suggerisco sempre di abbinare alt , width HTML5 e height HTML5

(per i motivi già descritti nel capitolo delle immagini).

Es. `<input type="image" src="freccia.jpg" alt="Submit" width="20" height="20" />`

(vedremo nell'esempio sottostante).

Per `<input type="submit" >` e `<input type="image" >`, sono validi anche gli attributi

- formaction HTML5,
- formenctype HTML5,
- formmethod HTML5,

formnovalidate HTML5,

formtarget HTML5.

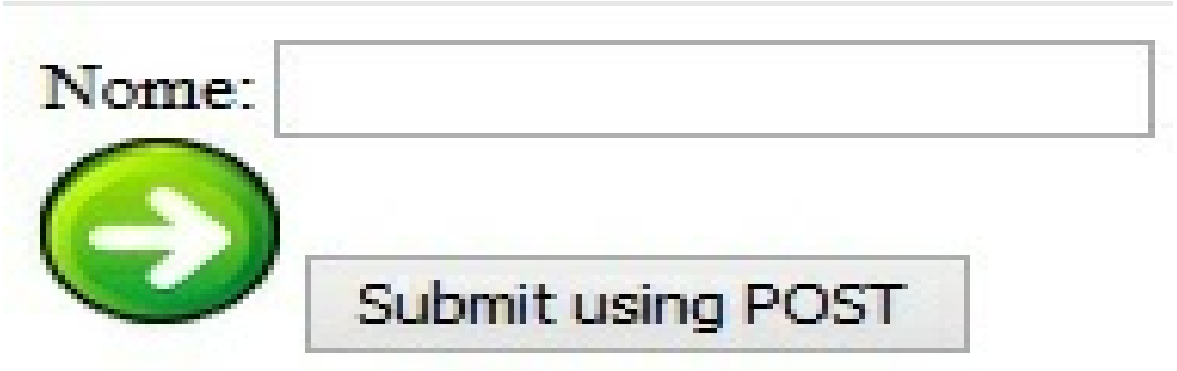
Nel momento in cui nel tag <form> si pone una determinata scelta per l'attributo action, method e gli altri non c'era possibile di dare più scelte, per esempio se volevamo lasciare all'utente di inviare i dati con GET o POST, non era possibile creare due bottoni, una per l'invio con GET e uno per l'invio con POST ma adesso lo è

Esempio di form nel cui tag <form>

spedisce i dati con GET ma si crea un altro bottone per inviare i dati in POST.

```
<form action="processaDati_get.php"
Nome: <input type="text" name="n
<input type="image" src="freccia.jpg'
<input type="submit" value="Submit
</form>
```

che risulta essere:



e anche se dall'immagine e con il solo HTML non si può capire la differenza, cioè permette di ottenere differenti metodi di invio in base alla scelta dell'utente.

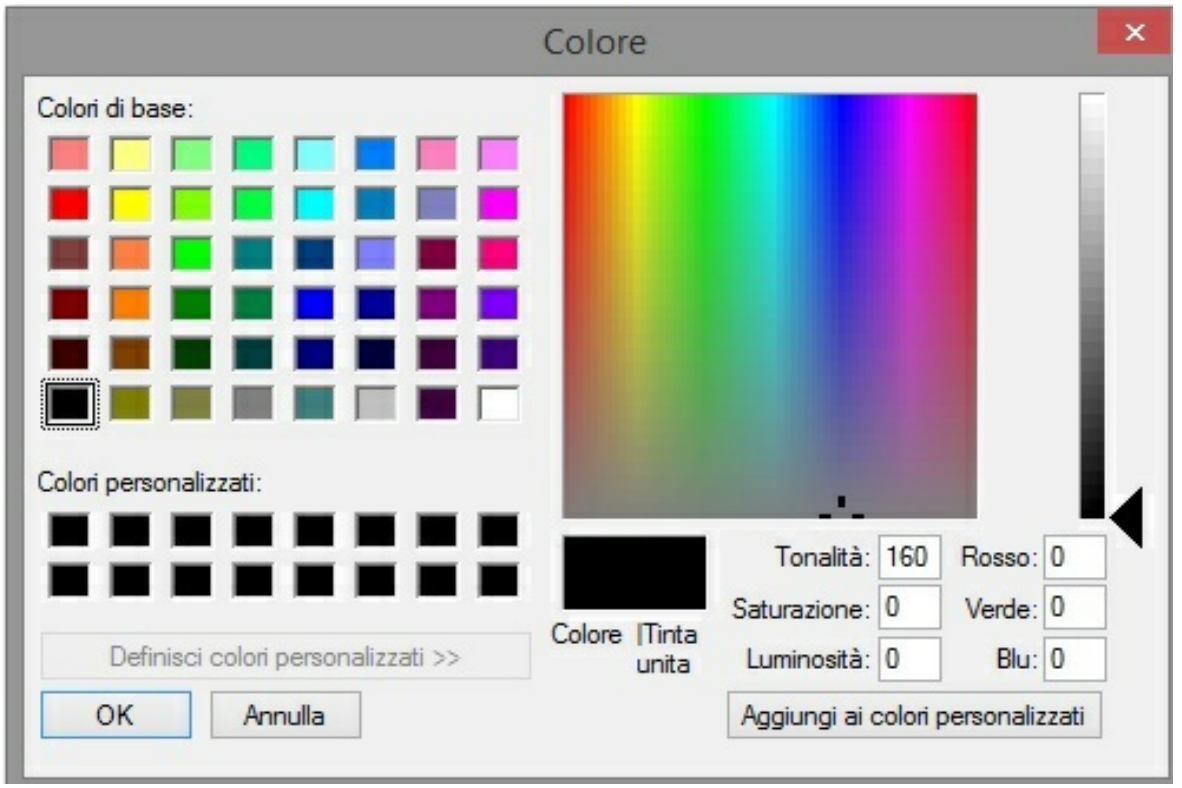
9.7 Gli Input Type di HTML5

In questo paragrafo mostriamo tutti gli <input type > introdotti a partire da HTML5.

9.7.1 color

Permette di aprire una finestra e selezionare dalla tavolazza il proprio colore.

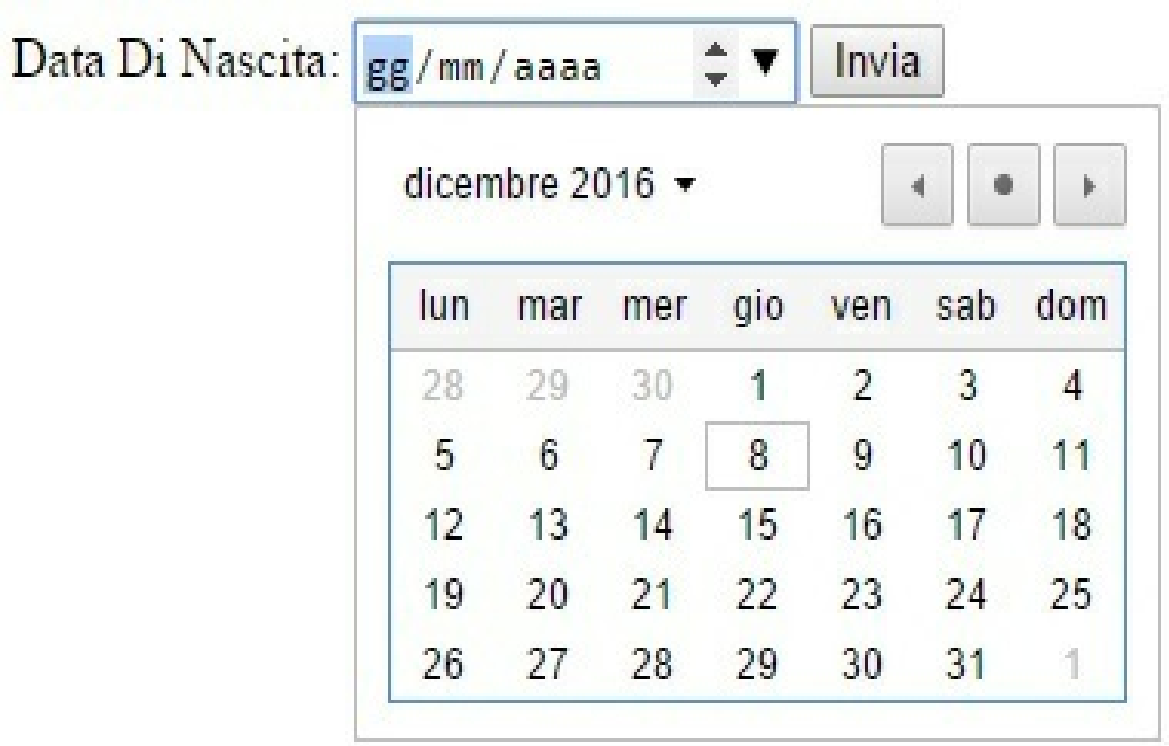
```
<input  type="color"  />
```



9.7.2 date

Permette di aprire una finestra e selezionare la data (non supportato da Firefox).

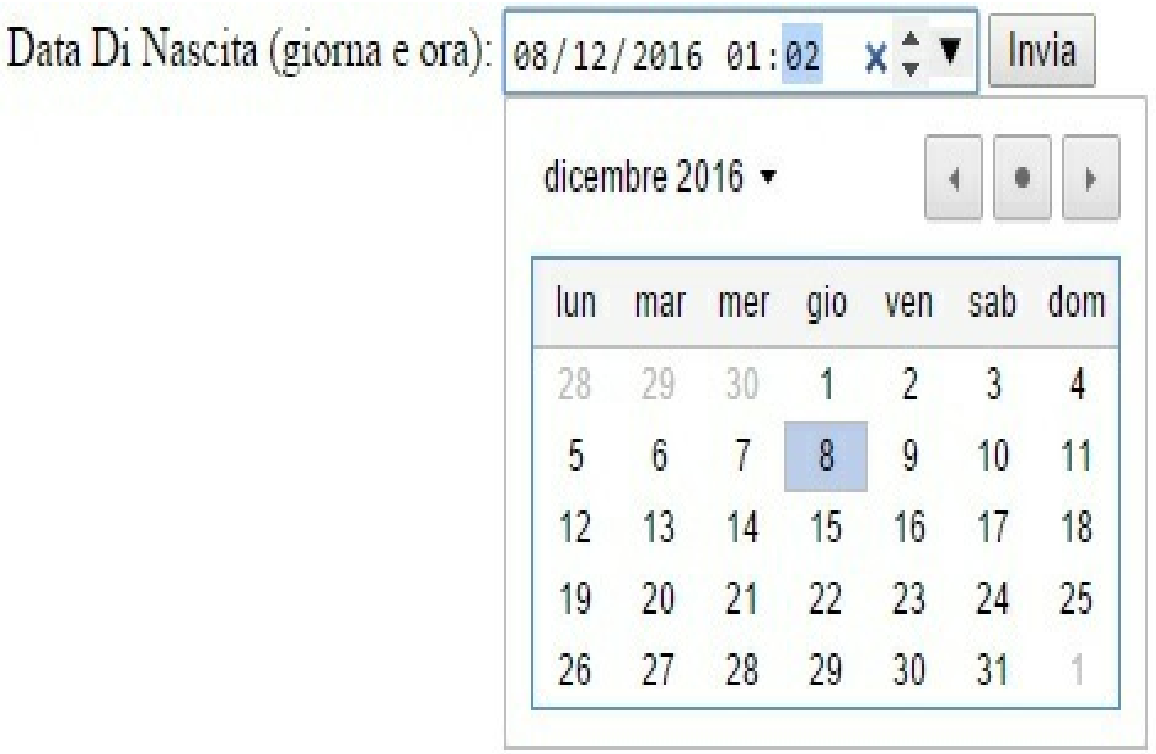
```
<input  type="date"  />
```



9.7.3 datetime-local

Permette di aprire una finestra e selezionare la data e l'ora (non supportato da Firefox).

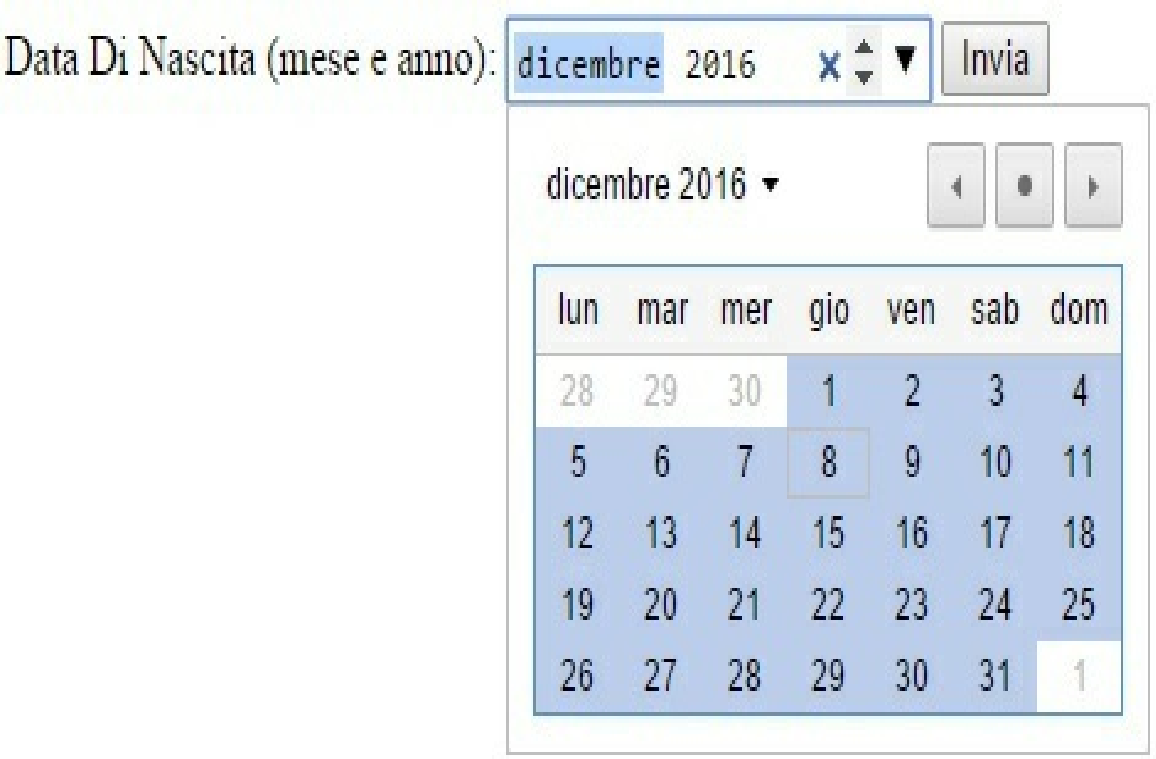
```
<input  type="datetime-local"  />
```



9.7.4 month

Permette di aprire una finestra e selezionare la data e il mese(non supportato da Firefox).

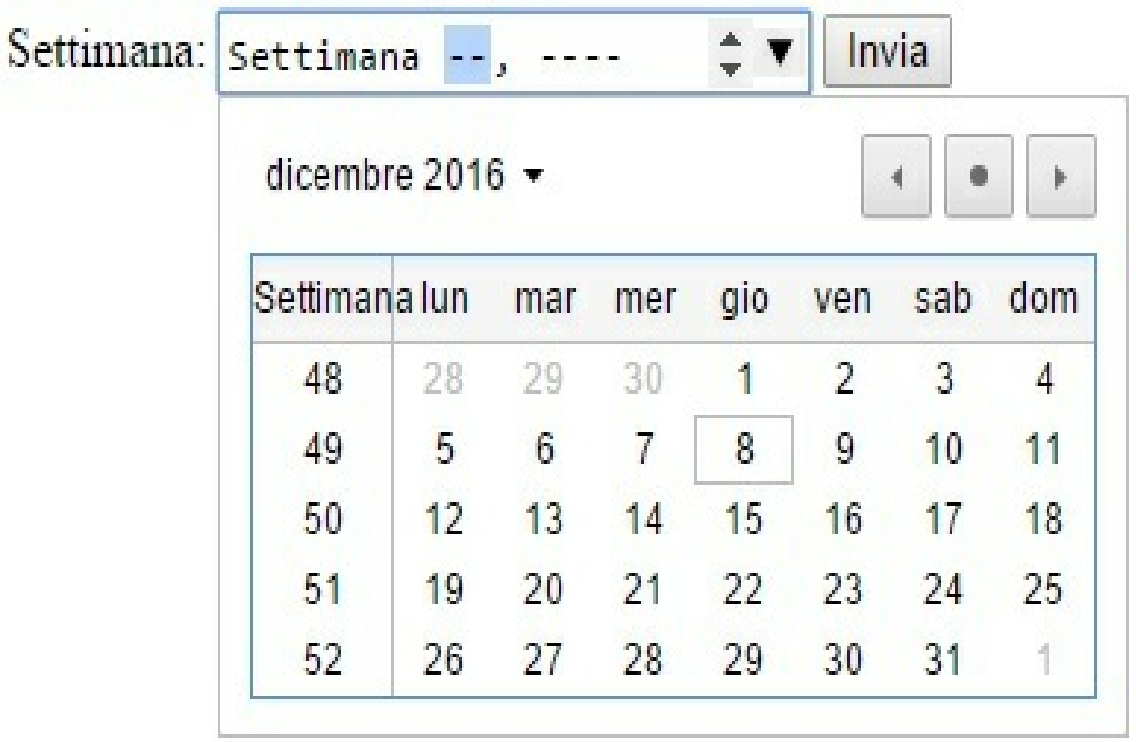
```
<input  type="month"  />
```



9.7.5 week

Permette di aprire una finestra e selezionare la data e il mese(non supportato da Firefox).

```
<input  type="week"  />
```



9.7.6 time

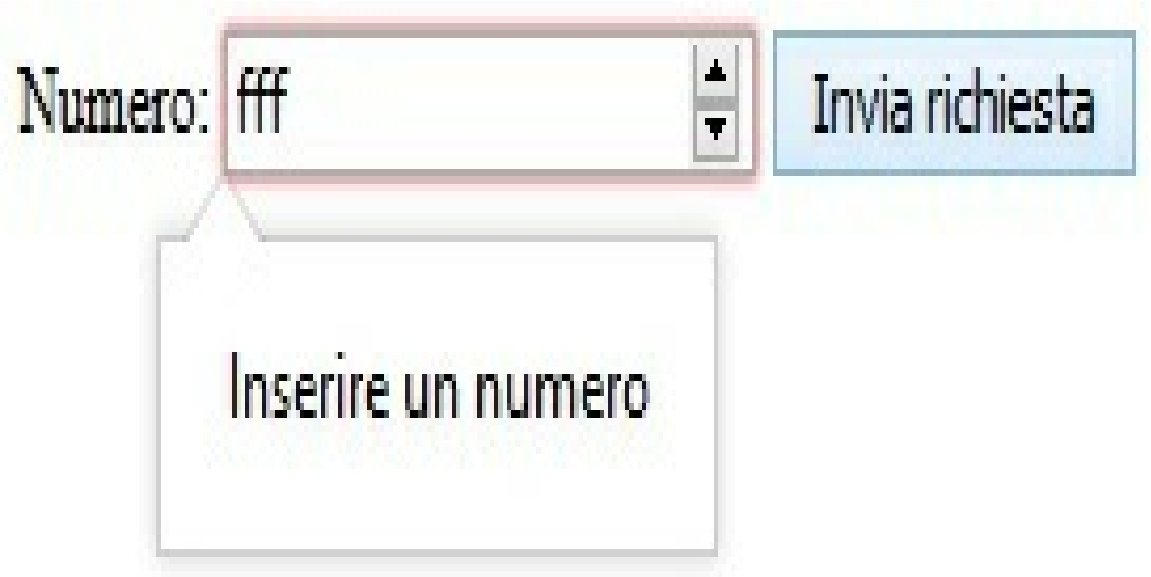
Permette di aprire una finestra e selezionare l'orario (non supportato da Firefox).

```
<input  type="time"  />
```

9.7.7 number

Casella di testo che obbliga l'utente a inserire un numero.

```
<input  type="number"  />
```

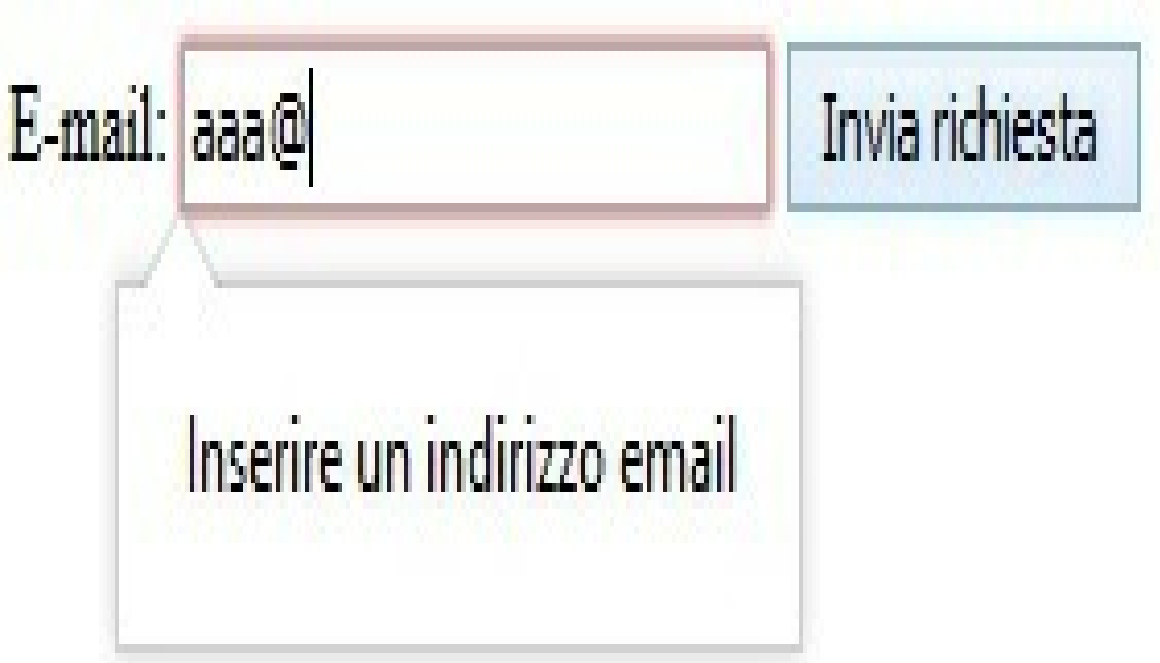


9.7.8 email

Casella di testo che obbliga l'utente a inserire una qualche stringa prima e dopo il simbolo di @. E' possibile inserire più email utilizzando multiple = "multiple" e separando ogni email scritta all'interno della

casella di testo tramite una virgola.

```
<input  type="email"  />
```



9.7.9 search

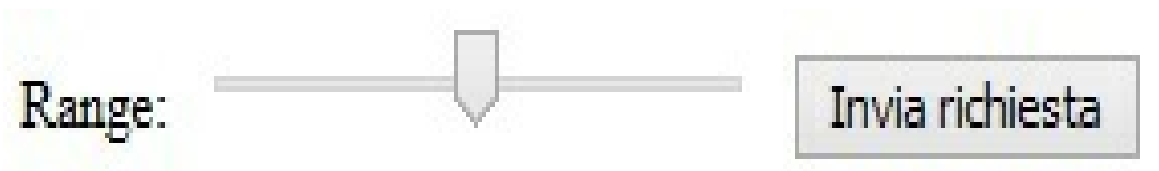
Casella di testo - solo in Safari viene mostrata coi bordi arrotondati.

```
<input  type="search"  />
```

9.7.10 range

Slide per scegliere un valore numerico, da utilizzare con min e max .

```
<input  type="range"  min  ="1"  e  ma
```



Esistono anche type = "url" e

type = "tel" ma sono supportati solo da alcuni browser. Come avrete notato, nel resto dei capitoli ho evitato di fornire istruzioni su tag o attributi presenti solo in alcuni browser perchè il nostro obiettivo è che il sito sia disponibile a tutti in maniera perfetta. C'è stata eccezione per questo paragrafo, però non descrivo proprio "tel" e "url" poichè il primo disponibile solo in Safari e il secondo in Safari e IE (nuove versioni).

9.8 Associazione Attributo - Type

In questo paragrafo indichiamo per ciascun attributo a quale valore di type è associabile.

Attributo	Type
accept	file
multiple	file, email
alt, height, width, src	image
formaction,	

formenctype, formmethod, formtarget	image, submit
checked	radio, checkbox
accept	file
placeholder, size	text, search, url, tel, email, password
autocomplete	quelli indicate per placeholder + date picker, range, color
required	quelli indicate per placeholder + date

	picker, number, checkbox, radio, file
step, min, max	number, range,date, datetime, datetime-local, month, time, week
autofocus, disabled, name, readonly	valido per tutti

9.9 Raggruppamento Controlli

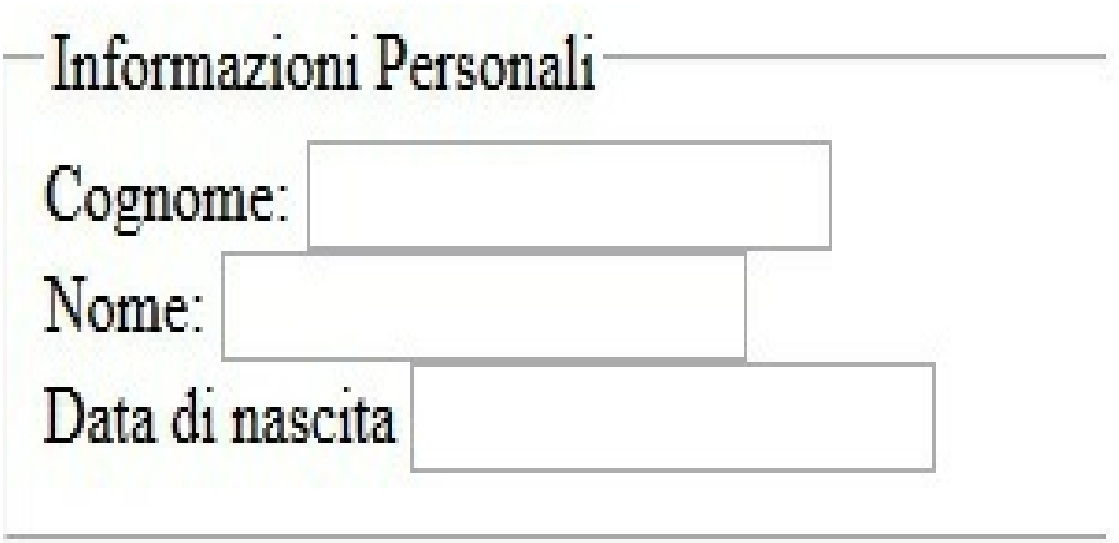
E' possibile raggruppare visivamente i controlli in un riquadro attraverso <fieldset> .Per dare un titolo si usa <legend> .

Gli attributi di <fieldset> sono name = "valore" HTML5 , disabled HTML5 e form = "valore" HTML5 (attualmente non supportato da nessun browser).

Es.

```
<fieldset>
<legend> Informazioni Personali
Cognome: <input type="text" name
Nome: <input type="text" name="n
Data di nascita: <input type="te
</fieldset>
```

che risulta essere:



9.10 Riepilogo

Abbiamo visto finora cosa sia HTML, permettendo di strutturare documenti attraverso i tags.Abbiamo analizzato come inserire link esterni, immagini ecc. Infine nel capitolo appena concluso si è trattato la preparazione di moduli per permettere l'inserimento dei dati da parte dell'utente.

Non abbiamo però visto come trattare questi dati poichè dobbiamo richiamare uno script indicato

nell'attributo action della form .

Esamineremo a breve la gestione lato server.

10. Tags HTML5 per strutturare una pagina

10.1 Tag section

Per rappresentare una generica sezione di un documento, cioè una serie di contenuti accomunati da un argomento, si utilizza <section> e per ogni sottosezione si utilizza uno dei tag per le intestazioni (<h1> ,- <h6>). Es. Supponiamo di voler descrivere Firefox, per ogni

caratteristica dedichiamo una sottosezione, ma se esse fossero prese singolarmente non avrebbero molto senso.

```
<section>
<h1>Firefox</h1>
<p>Permette di visualizzare dc
</section>
<section>
<h1>Logo</h1>
<p>Mozilla Firefox ha come log
</section>
```

risulta in output:

Firefox

Permette di visualizzare documenti HT

Logo

Mozilla Firefox ha come logo …

10.2 Tag article

Qualora il contenuto descritto può essere presentato in maniera indipendente, meritando un proprio articolo si utilizza <article> , che possiamo inserire in <section> , poichè come detto <section> è un contenitore di argomenti correlati, <article> specializza la <section> .

Es: Parlo dei vari Browser, tra cui Firefox, IE ecc..

```
<section>
<h1>Browsers</h1>
<p>Permettono di visualizzare
<article>
<h2>Firefox</h2>
<p>Mozilla Firefox è nato ...<
</article>
<article>
<h2>Explorer</h2>
<p>Internet Explorer è nato ..
```

```
</article>
</section>
```

risulta in output:

Browsers

Permettono di visualizzare documenti

Firefox

Mozilla Firefox è nato …

Explorer

Internet Explorer è nato …

10.3 Tag header

Qualora si voglia fornire un gruppo di informazioni introduttive o di aiuto alla navigazione si può utilizzare il tag <header> , che oltre a contenere i tag per le intestazioni, può contenere un logo, un menù navigazionale, informazioni sull'autore della pagina e

cosi via...

```
<article>
<header>
<h1>Programmare per Il Web</h1
<h3>HTML5, CSS3, PHP+MySQL, ..
</header>
<p>Il libro tratta di ...</p>
</article>
```

risulta in output:

Programmare per Il V

HTML5, CSS3, PHP+MySQL, ...

Il libro tratta di ...

10.4 Tag nav

Abbiamo parlato della possibilità di inserire dei menù all' interno dell' <header> , il tag per far ciò è <nav> . Possono esserci più tag

<nav> all'interno di una stessa pagina, per esempio uno per rappresentare la navigazione all'interno dell'intero sito e uno per esplorare i contenuti della pagina corrente.

Es. di menù con link per raggiungere la Home, Chi Siamo, I contatti.

```
<nav>
 <ul>
 <li><a  href="home.html">Home</a><
 <li><a  href="chiSiamo.html">Chi Si
 <li><a  href="Contattaci.html">Conta
```

```
  </ul>
</nav>
```

risulta in output:

- **Home**
- **Chi Siamo**
- **Contattaci**

10.5 Tag main

Il tag <main> specifica il contenuto principale del documento, che dovrebbe essere unico all'interno del documento. Non dovrebbe contenere

le informazioni tipiche del <footer>

10.6 Tag footer

Contiene informazioni sull'autore, di copyright, contatti, sitemap ecc. Molto comune nei siti odierni.

10.7 Tag aside

Contiene informazioni secondarie rispetto al contenuto principale, che di solito vengono poste nei lati della pagina.

10.8 Tag figure, figcaption

Per dichiarare un diagramma, snippet (parti) di codice, immagini che rappresentano contenuto indipendente si utilizza <figure> , opzionalmente in congiunzione con <figcaption> per aggiungere una didascalia.

```
<figure>
 <img  src="pallina.png"  alt="pallina"
<figcaption> Una pallina da tenn
```

```
</figure>
```

risulta in output:

Una pallina da tennis.

10.9 Tag details,

summary

Il tag <details> usato con <summary> permette di dare una serie di informazioni, ma, invece di mostrarle tutte, viene mostrato solo il testo indicato nel sommario e l'utente dovrà premervi per mostrare tutti i dati. Supportato solo dalle ultime versioni dei browsers e non supportato da Internet Explorer, al momento in cui scrivo (fine 2016), sconsiglio di utilizzarlo perchè solo una piccola percentuale di utenti

aggiorna all'ultima versione il software.

```
<details>
 <summary> Programmare per il W
   <p> - Ciro Ragone. Tutti i d
   <p>HTML5, CSS3, PHP-MySQL, .
 </details>
```

risulta in output:

▶ **Programmare per il WEB**

10.10 Altri Tags

10.10.1 Tag meter

Per indicare graficamente un valore all'interno di un range, possiamo utilizzare il tag <meter> (non supportato da IE). Gli attributi supportati sono min e max per indicare un valore minimo e uno massimo. Se non settati il range si intende compreso tra 0 e 1. Se è solo il minimo a non essere settato il suo valore è automaticamente settato a 0. Per indicare il valore corrente si usa invece value .

```
Il voto medio della classe è r
```

risulta in output:

```
Il voto medio della classe è r
```

Come si può notare, il colore è verde ma volendo possiamo anche stabilire dei valori "critici" attraverso gli attributi low e high. Il primo deve essere maggiore di min ma minore di max e high, mentre il secondo deve essere maggiore di min e low ma minore di max.

Se volessimo ad esempio rappresentare la seguente situazione:

la temperatura minima deve essere di 30, il massimo di 100, i valori critici sono rappresentati da tutti quei valori al di sotto del 70 o al di sopra di 90, allora scriveremo:

```
Stato Temperatura attuale: <m
```

risulta in output:

Stato Temperatura attuale:

```
Stato Temperatura attuale: <m
```

risulta in output:

Stato Temperatura attuale: 

Se, infine, volessimo indicare un valore ottimale si utilizza optimum .

10.10.2 Tag progress

Se <meter> è utilizzato per rappresentare una misura, <progress> è utilizzato per rappresentare la percentuale di completamento di un task, quindi viene utilizzato in congiunzione con JavaScript.

Presenta due soli attributi, value e max i quali rappresentano rispettivamente la percentuale di

completamento del task e quanto lavoro viene richiesto in totale.

```
Il lavoro è stato completato a
```

risulta in output:

```
Il lavoro è stato completato a
```

10.10.3 Tag ruby, rt, rp

Per mostrare come si pronunciano caratteri estasiatici si usa il tag <ruby> in congiunzione con <rt> che effettivamente definisce la pronuncia del carattere e <rp> nel

caso in cui il tag <ruby> non sia supportato.

```
<ruby>
   ? <rp>(</rp><rt>Kan</rt><rp> )</
   ? <rp>(</rp><rt>ji</rt><rp>)</rp
   </ruby>
```

risulta in output:

Esempio di ruby

Non discuteremo dei seguenti tags perchè non supportati dai maggiori browsers o utilizzati per

scopi specifici:

<dialog>

<keygen>

<bdi>

<menu> **e** **<menuitem>**

.

Evitiamo anche il tag <time> perchè non viene rappresentato in maniera diversa dal normale testo.

Infine, anche se

molto
interessante, non
descriviamo
<datalist>
perchè non
supportato da
Safari e
l' <autocomplete>
che descriveremo
con JS.
Infine, tratteremo
la parte grafica
(canvas, svg,
mostrare una

mappa utilizzando Google) e le API al momento di JavaScript.

1.

INTRODUZIONE

PHP è l'acronimo di "PHP HyperText Preprocessor", inizialmente era l'acronimo di "Personal Home Page". E' un linguaggio di programmazione creato da Rasmus Lerdorf nel 1994

per costruire delle estensioni in documenti HTML e migliorare così la sua home page personale.

E' per eccellenza il linguaggio lato server.

Il codice scritto in tale linguaggio viene processato dal server, il quale estrapola i risultati

e risponde alle richiesta

La curva di apprendimento è brevissima, soprattutto se si conoscono già i concetti fondamentali della programmazione strutturata.

In questo testo esploreremo tutti le keyword con i

relativi esempi ma qualora si volessero approfondire i concetti relativi alla programmazione con i costrutti essenziali è possibile far riferimento a PROGRAMMAZIONE STRUTTURATA: TECNICHE DI

PROGETTAZIONE E PROGRAMMAZIONE (C e Pascal). A breve proporrò anche un libro per la programmazione in PHP mettendo in evidenza i costrutti.

2.

PROGRAMMARE IN PHP

2.1

Hello,World

Su Windows, creiamo un "Nuovo Documento di Testo" cliccando con il tasto destro

del mouse e scegliendo dal menù Nuovo > Documento di Testo

Come nome del documento scegliamo Hello e salviamo con estensione .php

A questo punto, scriviamo il nostro codice:

```
<html>
<head></head>
<body>
<?php
echo ("Hello,World");
?>
</body>
</html>
```

Come si può notare, è possibile scrivere codice PHP integrato in quello HTML.

Per indicare all'interprete che ciò che scriviamo deve essere inteso come PHP, si inizia il codice con <?php e si conclude con ?>.

Alternativamente si può usare

<script language = "php" >

concluso con

</script> o

ancora più semplicemente <% ... %> o <? ...?> (ma va abilitato nel file *php.ini*).

Il primo modo è da preferire agli altri perchè permette la costruzione di documenti *xml e xhtml*

La parola chiave per permettere di

stampare una frase all'interno del documento è echo . La stringa che segue deve essere circondata dalle virgolette.

Per provare quanto scritto è necessario aver installato XAMPP, dopo averlo fatto accedete alla cartella htdocs che

in Windows è localizzata in C:\xampp\htdocs e copiate il file Hello.php all'interno

A questo punto aprite Firefox(o il vostro browser predefinito) e digitate http://localhost/hello.php

Dopo aver premuto il tasto

invio comparirà a video Hello,World Come si può notare otteniamo proprio ciò che ci aspettavamo, il client richiede la pagina hello.php e il server dopo aver eseguito il codice, lo inserisce nel flusso html e invia la risposta al richiedente

3.

VARIABILI

3.1

Dichiarazione

Nel PHP, a differenza di molti altri linguaggi (es. C e Java), non c'è necessità di dichiarare il tipo di

variabile, questa verrà assegnata in fase di esecuzione.

Per utilizzare una variabile basta utilizzare il simbolo $ seguito dal nome della variabile.

Ad esempio, se io volessi utilizzare una variabile di nome test con valore 5 all'interno del programma

dovrei scrivere

```
$test=5;
```

3.2 Tipi Di Variabili

E' possibile avere variabili di tipo boolean, integer, float e string .

Per il primo tipo possiamo dare valore TRUE o

FALSE (case unsensitive).

Per il secondo tipo valori interi quindi appartenenti all'insieme Z (-3,-2,...,2,3...)

Per il terzo tipo si considerano anche i valori decimali (3.2,2.3,...)

Infine le stringhe vengono circondate da

doppie virgolette o
da apice quindi
sono
siffatte:*"stringa"* o
'stringa'.
Per definire
stringhe su più
righe si utilizza

```
$stringa=<<<keyword  LaStr
```

Per capire meglio
segue un esempio:

```
$stringa=<<<MiaStringa
```

Questa
la
scrivo
su più righe **MiaStringa**;

E' possibile inoltre
dichiarare oggetti,
array, risorse,
valori null ma
verranno trattati
all'occorrenza,
ognuno nel proprio
paragrafo.

3.3 Settare le variabili

Per settare il tipo di variabile è possibile utilizzare settype oppure cast .

Per il primo metodo bisogna scrivere

```
settype ($variabile,"tipo")
```

Per il secondo

metodo basta scrivere $var=((tipo) $variabile).
Quest'ultimo è da preferire, perchè presente anche in molti altri linguaggi.
Effettuare questo tipo di conversione è utile quando ad esempio abbiamo un numero

decimale e
vogliamo sia
troncato oppure
abbiamo una
stringa composta
da lettere e numeri
che, convertita in
intero, restituisce
solo i numeri

3.4
Identificare

le variabili

Può essere necessario capire di che tipo sia la variabile, ad esempio quando vogliamo validare l'input.

Per capire di che tipo sia la variabile, durante la fase di debug, si può utilizzare

settype ($variabile).

Se vogliamo invece verificare se la variabile sia di un certo tipo si usa *is_tipo*

Per esempio, se volessimo verificare che il valore inserito in input sia un numero intero si può utilizzare

is_int ($variabile).

Segue lo stesso ragionamento per gli altri tipi (is_bool,is_float,is_string,is

Per verificare se una variabile è valorizzata si usa

isset ($variabile)

Queste istruzioni vanno inserite all'interno di blocchi condizionali (if..else), quindi

successivamente faremo degli esempi

3.5 Costanti

Per dichiarare delle costanti è necessario scrivere

```
DEFINE ("key","value")
```

Si noti che non è possibile usare il valore di variabili come value.Non è

ammissibile

scrivere

DEFINE ("VALORE","CIAO $var") anche se "$var" è stato definito precedentemente.

4. STAMPA

4.1 Echo

Abbiamo già accennato a come stampare delle semplici frasi a video nel paragrafo 2.1.

C'è la possibilità di usare diverse keywords.

Analizziamo echo .

Tramite questa parola chiave è possibile stampare semplice frasi, oppure testo, contenuto di variabili e tag html Supponendo di avere una variabile chiamata $test valorizzata a 5, è possibile quindi scrivere:

```
<html>
<head></head>
<body>
<?php
$test=5;
echo "Il valore della variabile
?>
</body>
</html>
```

Come si può notare, è molto comodo stampare valori in questo

modo.

Si tenga presente che se si usa l'apice singolo *'..'* e non le doppie virgolette*"..."* non verrà stampato il valore della variabile ma ciò che è stato scritto.

Es: echo '$test' stampa $test mentre echo "$test" stampa 5

Alternativamente, sempre con echo è possibile sfruttare l'operatore di concatenazione . (il punto) per poter manipolare le variabili contemporaneamente alla stampa senza modificarne il contenuto.
Supponenendo di

avere due variabili denominate var1 e var2, vogliamo addizionare i valori e stamparli direttamente

```
<html>
<head></head>
<body>
<?php
$var1=5;$var2=3;
echo "Il valore della variabili
?>
```

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

Si noti che la è accentata è stata scritta come indicato per gli special char poichè l'output è poi stampato nell'html. Per andare a capo si utilizza `<br/>` che viene identificato e

interpretato come tag.

IMPORTANTE: Per inserire valori tra virgolette è necessario usare backslash \" .

Es. echo "<font color= \" blue \" >" producendo in output <font color = "blue" > .

4.2 PHP in HTML

Altro modo interessante per stampare caratteri e meglio ancora, HTML complesso si può 'mischiare' PHP e HTML.

Segue un piccolo esempio, ma questa scrittura è molto comoda

quando si vuole stampare tabelle o tanto testo HTML.

```
<html>
<head></head>
<body>
<?php
$var1=5;$var2=3;
Il valore della variab
?>
</body>
</html>
```

4.3 Print

Permette di stampare frasi e variabili. La principale differenza della echo è che impiega più tempo

```
<html>
<head></head>
<body>
<?php
$var1=5;$var2=3;
```

```
print  Il valore della variabili
?>
</body>
</html>
```

4.4 Printf

Permette di stampare frasi e variabili formattando l'output

```
<html>
<head></head>
```

```
<body>
<?php
$var1=5;$var2=3;
printf  ("%1.2f",1/6);
?>
</body>
</html>
```

5.

OPERATORI

5.1

Operatori Matematici

OPERATORE	SINTASSI
Negazione	-$var1
Addizione	$var1+$var
Sottrazione	$var1-$var2

Moltiplicazione	$var1*$var
Divisione	$var1/$var
Modulo	$var1%$var
Concatenazione	$var1.$var2 (per stringhe)
Esponente	$var1**$va (da PHP 5.6

L'operatore modulo restituisce il resto di una

divisione. Es.
10%3=1 poichè
10:3 ha resto 1

L'operatore
divisione
restituisce un float
a meno che
entrambi i numeri
siano di tipo intero

5.2
Operatori di

Assegnazione

E' possibile assegnare ad una variabile in questo modo $a=$b+$c-$d...., ovvero l'espressione a destra dell'uguale è valutata e il valore finale salvato in $a.

Si ricorda che la variabile presente

a sinistra può anche essere presente a destra. Questa tecnica è molto utilizzata infatti il linguaggio supporta la sintassi seguente:

OPERATORE	SIGNIFIC
$var1=$var1+$var2	$var1+=
$var1=$var1-$var2	$var1-=
$var1=$var1*$var2	$var1*=

$var1=$var1/$var2	$var1/=
$var1=$var1%$var2	$var1%=
$var1=$var1.$var2	$var1.=
$var1=$var1+1	$var1++
$var1=$var1-1	$var1--/

La differenza nell'usare $var1++ e ++$var1 consiste nell'ordine delle operazioni. Nel primo caso il

valore verrà utilizzato e poi incrementato, nel secondo usato accade il viceversa

```php
<?php
$var1 =  3;
echo "Valore:".$var1++; //sta
echo "Valore:".$var1; //stam
echo "Valore:".++$var1; //in
echo "Valore:".$var1; //stam
?>
```

5.3 Operatori Relazionali

Nei linguaggi di programmazione, una classe di elementi spesso utilizzata per la costruzione degli algoritmi è quella degli operatori relazionali.

OPERATORE	SIGNIFICATO
<	Minore Di
<=	Minore o Eguale Di
>	Maggiore Di
>=	Maggiore o Eguale Di
==/===	Uguale a/Identico a (valore e tipo)
	Diverso

!=,<> / !==	da/Non Identico

Es. $a=3; $b="3";

$a==$b restituisce TRUE mentre

$a===$b restituisce FALSE

5.4 Operatori Logici

Questi operatori permettono di definire le condizioni. Sono tre e rispecchiano la cosiddetta Algebra di Boole.

5.4.1 AND

L'operatore AND, indicato con AND o && specifica che la condizione è soddisfatta se e

solo se tutte le condizioni prese singolarmente sono soddisfatte.

Es. Ho i soldi **E** ho il tempo libero, perciò vado al cinema.

Posso andare al cinema solo perchè entrambe le condizioni sono soddisfatte contemporaneamente.

Nella tabella seguente analizziamo la tavola della verità AND. Con 0 indichiamo la condizione non soddisfatta e con 1 il soddisfacimento.

A	B	A && B
0	0	0
0	1	0
1	0	0

1	1	1

Come detto A&&B è vero se e solo se A è vero **E** B è vero.

Es. *informatico.*

Vogliamo verificare che un numero sia compreso tra 3 e 10 (estremi inclusi).

N>=3 && N<=10

5.4.2 OR

L'operatore OR, indicato con OR o || , specifica che la condizione è soddisfatta se e solo almeno una delle due è soddisfatta. Es. Mangio il gelato **O** mangio la pizza.

A	B	A || B
0	0	0
0	1	1

1	0	1
1	1	1

Come detto **A||B è vero se almeno una è vera.**

Es. *informatico*

Vogliamo verificare che un numero sia minore di 3 o maggiore di 10 (estremi inclusi).

Ciò va scritto come

N<=3 || N>=10.

5.4.3 NOT

L'operatore NOT, indicato con !, nega l'espressione.

Es. A=Ho i soldi; !(A)= Non ho i soldi;

A	!(A)
0	1
1	0

Es.

*informatico.*Vogliamo verificare che un numero non sia minore di 3 o maggiore di 10 (estremi inclusi).
Ciò va scritto come
!(N<=3 && N>=10)

5.4.4 XOR

L'operatore XOR, indicato con XOR, specifica che la

condizione è soddisfatta se e solo una delle due è soddisfatta. Es. Mangio il gelato **ma non** mangio la pizza.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Come detto **A XOR B è verificata se sola una è vera.**

6.

INCLUDERE FILE ESTERNI

Potrebbe capitare di dover usare librerie esterne o noi stessi potremmo vorremo creare dei file nei quale

inserire le operazioni più comuni che ripetiamo sempre oppure per tenerare separare diversi concetti che compongono alla fine un'entità più complessa.

Per far ciò si usano due keyword:

include e

require .

La differenza tra le due consiste nella scelta del comportamento nel caso in cui il file richiesto non sia disponibile, in particolare con include qualora il file sia assente si ha solo un warning mentre con require non si permette la

prosecuzione del programma

Per evitare che variabili, dati e funzioni vengano caricate più volte (poichè un file potrebbe richiamare A.php e B.php e A.php potrebbe richiamare B.php) si utilizza la versione

include_once e require_once.

La sintassi per inserire file esterni (da copiare nel top del codice) è una delle seguenti:

```
<?php
include "path" ;
include_once "path" ;
require "path" ;
require_once "path" ;
?>
```

7.

SELEZIONE

7.1 Selezione Binaria

Tramite questo strumento è possibile effettuare delle azioni se una condizione è

verificata e qualora non lo fosse si eseguono altre determinate operazioni o si può anche non fare nulla.

Analizziamo il codice in cui una sola azione è eseguita.

```
<?php
if (<condizioni>)
```

```
    { blocco_azioni }
?>
```

Analizziamo il codice in cui se la condizione non è soddisfatta un blocco alternativo viene eseguito.

```
<?php
if (<condizioni>)
    { blocco_azioni }
else
    { blocco_azioni }
```

```
?>
```

Alternativa più compatta all'*if..else* è la seguente:

```
<?php
(<condizioni>)?  azione
?>
```

ESERCIZIO: Se il valore della variabile è pari si stampi la metà altrimenti il

doppio.

```
<?php
if ($N==0)
    echo   "La metà di $N è "
else
    echo   "Il doppio di $N è
?>
```

Suggerimento: Si noti che istruzioni contenute sia nell'if, sia nell'else, molto probabilmente

devono essere eseguite a prescindere, quindi non dovrebbero essere soggette a condizioni

7.2 Selezione Multipla

Supponiamo di voler valutare un

test in base al numero di risposte corrette

PUNTEGGIO	VALUTAZIONE
0..1	Insufficiente
2..3	Sufficiente
4..5	Buono
6..7	Ottimo

Attraverso lo switch scriviamo molto codice che riduce la leggibilità dello stesso. Per ovviare a questo problema possiamo utilizzare la **SELEZIONE MULTIPLA**.

```
<?php
switch ($variabile){
case caso_1:
```

```
    { blocco_1;
     break ;}
case caso_2:
    { blocco_2;
     break ;}

default :
    { blocco_default;
     break ;}
?>
```

La keyword è

switch

($variabile). Ogni blocco di istruzioni questo deve essere racchiuso tra **{}** .Ogni blocco va concluso con break .

Da non sottovalutare la possibilità di definire un insieme di istruzioni qualora le precedenti non

abbiano successo attraverso la parola riservata default . Si noti che l'opposto di break è continue, che non interrompe il ciclo ma prosegue con l'iterazione successiva.

APPROFONDIMENTO:

Cosa succede se dimentichiamo di

scrivere il break al termine di un case ? Il break è una keyword che indica al compilatore di uscire dal blocco e di proseguire con il normale flusso di esecuzione. Se il case è omesso, allora tutti i casi successivi saranno eseguiti fino al

raggiungimento del termine o al raggiungimento di un break .

8.

ITERAZIONE

8.1 Iterazione Enumerativa

In questo capitolo ci approcciamo ad un concetto fondamentale dell'informatica.

Analizziamo i diversi metodi per implementare questo costrutto

```
<?php
for (init_variabili;  condizione
?>
```

C'è una variabile, che nel caso dei cicli in gergo viene detta indice, che assume un valore iniziale, e fino a

che non raggiunge il valore finale si esegue il gruppo di istruzioni indicate.
E' possibile definire il valore da aggiungere all'indice al termine di ogni iterazione (si definisce il passo).
Esercizio: Sommare i primi 10 numeri naturali.

In generale in informatica per gli indici si usano le variabili **i** e **j**

```
<?php
$N=10;
$somma=0;
for ($i=1;$i<=10;$i++)
echo "Somma finale $somma
?>
```

Non è necessario valorizzare nessuna delle tre

condizioni. E' quindi possibile scrivere for(;;) . Un ciclo di questo tipo però non avendo nessuna condizione potrebbe condurre ad un loop(ciclo) infinito dal quale uscire è impossibile a meno che non si utilizzi la parola chiave

break nel blocco di istruzioni del for cosi da interrompere il flusso nel blocco e ritornare alla normale esecuzione del codice. Questa soluzione non è elegante.
E' possibile inizializzare e incrementare più

valori contemporaneamente separando le istruzioni con la virgola.

Esempio inizializzazione variabili $i,$j e relativo incremento

```
for ($i=0,$j=0;$i<10;$i++,$j++)
```

8.2 Iterazione per Vero

Analizziamo ora l'iterazione per vero. Il blocco di istruzioni può anche non essere eseguito poichè in questo tipo di costrutto si esegue il codice del ciclo

superando la condizione iniziale.

```
<?php
$N=10;
$somma=0;
while (condizione)
    {istruzioni}
?>
```

Nell'iterazione per vero il blocco di istruzione viene eseguito fino al soddisfacimento

della condizione.

ESERCIZIO: Data una somma inizializzata a 0, si aggiunga alla suddetta un valore pari a 3 fino a quando il risultato non supera 50. Stampare le somme parziali

```
<?php
$somma=0;
```

```
while ($somma<=50)
   {$somma+=3;
    echo  $somma<br/>";}
}
?>
```

8.3 Iterazione DO..WHILE

Rispetto al caso precedente, almeno una volta il

blocco di istruzione viene eseguito

```
<?php
$somma=0;
do
   {$somma+=3;
    echo  $somma<br/>";}
}
while ($somma<=50)
?>
```

8.4 Equivalenza tra Costrutti

I tre metodi sopra analizzati permettono di effettuare la stessa operazione. Resta al programmatore decidere quale

costrutto sia migliore tra quelli proposti.

In genere il for (iterazione enumerativa) viene utilizzato se la condizione d'uscita è prefissata, il do..while è utile se si vuole ciclare almeno una volta infine il while (iterazione per

falso) si usa per testare fin da subito la condizione.
Qualche lettore potrebbe chiedersi come è possibile scrivere il problema precedente con un'iterazione enumerativa se non conosciamo la condizione

d'uscita?In aiuto giunge la keyword break

9. TIPI DI DATI STRUTTURATI

Fino ad ora abbiamo utilizzato solo tipi di dati semplici. In informatica nasce però la necessità di avere una collezione di oggetti dello stesso

tipo, che siano essi semplici o da noi definiti.

9.1 Vettori

Supponiamo di essere giudici di una gara alla quale partecipano 100 persone. Ad ognuna di questa viene assegnato un numero e un voto.

Dobbiamo quindi memorizzare 100 valori in 100 variabili diverse. Dovremmo dichiarare quindi partecipante0, partecipante1, partecipante2, .. Ciò ovviamente è un lavoro lungo e tedioso, che induce all'errore. Per ovviare a questo

problema nasce l'array (o vettore) che costituisce una lista ordinata di dati dello stesso tipo individuabili mediante la loro posizione nella lista stessa.

Il concetto di vettore informatico riprende il concetto di vettore matematico. In

questa disciplina tale nozione viene indicata con v_0, v_1,..,v_n. L'elemento di posto generico i viene indicato con v_i. Per accedere ad una determinata posizione i si utilizza la notazione <nome_vettore>[i]. E' possibile assegnare ad una

variabile generica X il valore di una cella del vettore, a patto che siano dello stesso tipo (X=A[i])

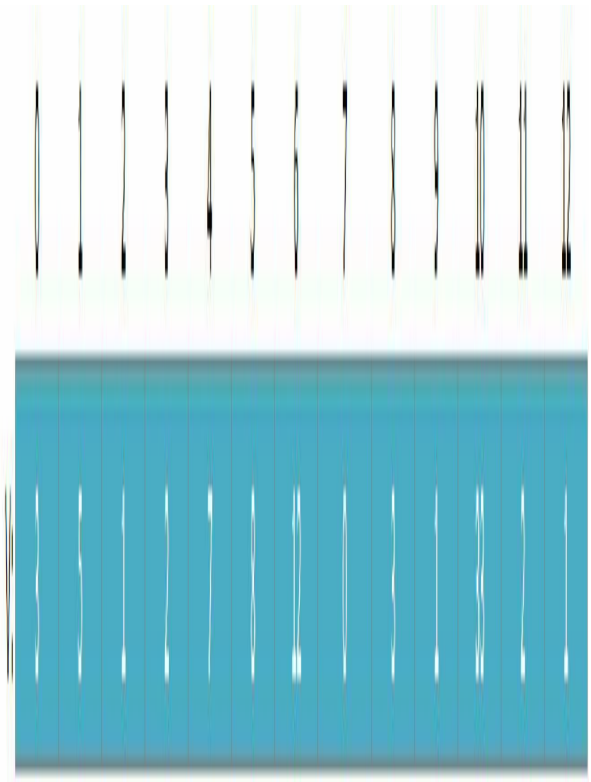


Figura 3.
Rappresentazione di un vettore

Suggerimento 1:

Nello specificare un indice per array, è possibile compiere operazioni matematiche direttamente nelle parentesi quadre

Suggerimento 2:

Non è possibile assegnare un vettore A ad un

altro vettore B, è vietato quindi scrivere B=A;

9.1.1 Inizializzazione

Dopo aver spiegato il concetto di vettore, analizziamo come utilizzare un array. Ricordando che non c'è necessità di

dichiarare la variabile, l'array può essere inizializzato in diversi modi:

```
<?php
$vett =  array ("val1","val2");
$vett[0]="val1";$vett[0]="val
$vett[]="val1";$vett[]="val2"
?>
```

E' possibile associare una chiave al valore

```php
<?php
$corso =  array ('prof1' => 'ma
$corso["prof1"]="materia1";
?>
```

9.1.2 Ciclo sull'array

Per sapere il numero di elementi contenuti nell'array si utilizza la funzione count ($array);

ESERCIZIO:

Stampare gli

elementi dell'array

$frutta=array1=>"mele",

2=>"pere",

3=>"banane").

```
<?php
$frutta=array(1=>"mele",
$cnt =count($frutta);
for ($i=1;$i<=$cnt;$i++)
echo $i,   ") ", $frutta[$i], "<b
?>
```

Altro modo è la

funzione

each ($array) che

inserita in un ciclo

restituisce la

coppia

(chiave,valore)

ESERCIZIO:

Stampare gli

elementi dell'array

$frutta=array("CIRO"=>"mele

"PINO"=>"pere",

"TINO"=>"banane").

<?php

```
$frutta=array(1=>"mele",
$cnt =count($frutta);
for ($i=1;$i<=$cnt;$i++){
$riga =  each ($frutta);
echo $riga["key"]  ," mangia
?>
```

Infine la funzione

foreach (array_expression as $key => $value)

che inserita in un ciclo restituisce la coppia (chiave,valore)

memorizzate nelle variabili $key e $value rispettivamente (che possono chiamarsi come ci pare)

ESERCIZIO:
Stampare gli elementi dell'array

```
$frutta=array("CIRO"=>"mele
"PINO"=>"pere",
"TINO"=>"banane").
```

```php
<?php
$frutta=array(1=>"mele",
$cnt =count($frutta);
foreach ($frutta  as  $key =>$
echo $key,   " mangia ", $valu
?>
```

9.3

Operatore

Array

OPERATORE	SIGNIFICATO

+	Unione delle due "liste"
==/===	Controlla se gli elementi delle liste hanno stessa coppia chiave/valore
===	Oltre al controllo dell'==, controlla che siano nello

	stesso ordine e delle stesso tipo
!=,<>	Controlla che non siano uguali
!==	Controlla che non siano identici

9.4 Stringhe

Argomento molto importante che merita un proprio paragrafo è la stringa.

Una stringa è una collezione di caratteri concatenati tra loro in modo da formare una parola.

Una stringa
termina sempre
con il carattere '\0'
(carattere
terminatore,
volgarmente detto
tappo).
La lunghezza
massima della
stringa è n-1
carattere
(l'ennesimo è per il
terminatore).
Qualora la stringa

dovesse essere più piccola le restanti posizioni restano vuote.

Essendo una stringa un vettore posso gestirlo accedendo ad una determinata posizione con la notazione A[i].

Le più comuni funzioni utilizzate sono:

- strlen ($stringa): restituisce la lunghezza di una stringa
- str_replace (da_sostituire,sostituisci): Sostituisce in $stringa la stringa indicata nel primo parametro con quello del

secondo.

- implode (delimitatore,$array): costruisce una stringa con gli elementi dell'array separati dal delimitatore.
- explode (delimitatore,$stringa): costruisce un vettore

considerando
per ogni
elemento
dell'array
l'elemento
della stringa
separato dal
delimitatore.

```php
<?php
$stringa =  "piece1 piece2 pie
$array =explode(" ", $strin
echo  $array[0]   // contiene
echo  $array[1]   // contiene
```

```
$nuova_stringa =implode(",
echo  $nuova_stringa   // = p
?>
```

10.

SOTTOPROGRAMI

Per risolvere problemi complessi conviene scomporli in moduli più piccoli. La parte principale di un programma richiama i **sottoprogrammi.** Questi ultimi, anche detti

subroutine,
vengono
richiamate da un
componente,
eseguono i calcoli
al loro interno,
restituiscono il
risultato (qualora
ci fosse) al
chiamante che
riprende la propria
esecuzione.
E' utile strutturare
un programma in

un sottoprogramma perchè ci permette di riutilizzare codice.
Supponiamo di dover scrivere tre volte sempre lo stesso codice in differenti punti, questo lavoro è lungo e tedioso, conduce all'errore, è difficilmente

comprensibile ed appesantisce la struttura del programma.
Per ovviare a questi problemi nasce il concetto di sottoprogramma.
I vantaggi sono i seguenti:
1. Scrivere più volte lo stesso blocco di istruzioni richiamando

semplicemente il nome del sottoprogramma

2. E' possibile applicare il blocco di istruzioni anche su variabili differenti, il che permette il riuso del codice

3. E' possibile integrare un programma già realizzato,

esprimerlo in sottoprogramma ed utilizzarlo in un altro programma. Abbiamo introdotto cosi un potente concetto informatico.

10.1 Chiamata a Sottoprogramma

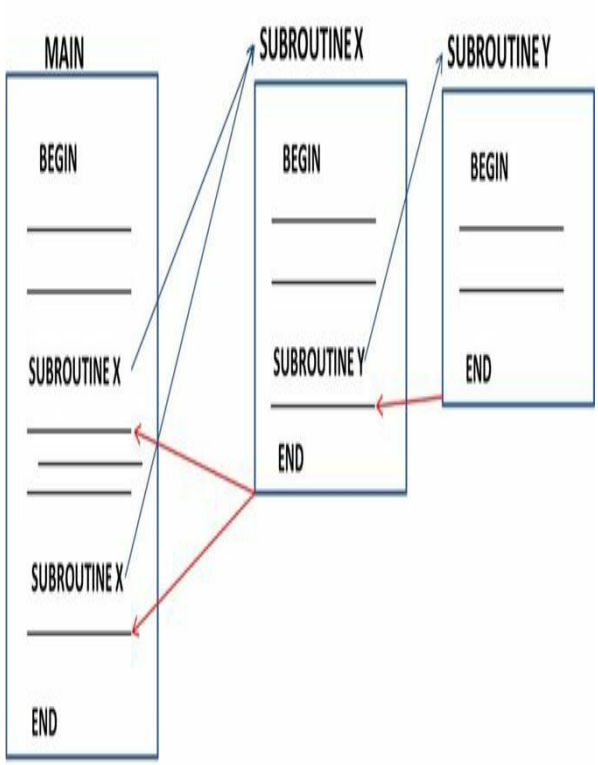


Figura 4.
Schema rappresentante chiamate a sottoprogrammi

Durante l'esecuzione del programma, si richiama una

subroutine denominata X, questa viene eseguita, il compilatore verifica la presenza di una subroutine Y, quindi la esegue. Terminato l'esecuzione del blocco di istruzioni contenuto in Y, il controllo viene ripreso da X, che

termina i propri calcoli quindi si continua con il flusso del main. Questo schema viene poi ripetuto una seconda volta con la successiva chiamata a X. In questo modo il programma diventa più leggibile, poichè non ripetiamo

sempre lo stesso codice in diversi punti.
E' possibile che il main invochi A che a sua volta invoca B che invoca C e così via. C'è però un limite a queste invocazioni infatti con un livello di invocazione troppo profondo il compilatore genera

errore.

10.2 Passaggio di parametri, Variabili Locali e Globali

All'interno di un

sottoprogramma si possono dichiarare costanti, tipi, variabili come proprio in un programma qualsiasi. Tutti i dati dichiarati all'interno di una subroutine vengono inizializzati al momento dell'**invocazione** e

vengono distrutti al momento della terminazione, inoltre hanno visibilità solo all'interno del sottoprogramma. Questo tipo di variabili vengono definite **locali.**

Se invece le variabili vengono dichiarate nel main sono dette **globali**

poichè hanno visibilità in tutto il programma e cessano di esistere solo con la terminazione del programma totale.
La potenza dei sottoprogrammi è data dalla possibilità di passare parametri.
Supponiamo di voler scrivere un

programma che scambi il valori di due variabili A e B.

```
<?php
$TEMP =  $A; //pongo il valo
$A =  $B; //valore di $B nella
$B =  $TEMP; //assegno il val
?>
```

Abbiamo utilizzato una variabile intermedia $TEMP e due variabili $A e $B. Supponiamo di

dover effettuare lo scambio tra altre due variabili C e D.

```php
<?php
$TEMP =  $C; //pongo il valo
$C =  $D; //valore di $D nella
$D =  $TEMP; //assegno il va
?>
```

E se avessimo altre due variabili E ed F?

Per risolvere questo problema è

permesso il passaggio dei parametri ovvero trasferire i dati dal chiamante al chiamato. Il sottoprogramma lavora con oggetti fantocci (**dummy parameters** "leggi dammi paramitars"), che sono variabili disponibili solo

all'interno del sottoprogramma e vengono usati come **parametri formali**. Essi non assumono valore fin quando non viene effettuata la chiamata vera e propria acquisendo cosi significato col programma chiamante. All'invocazione

avviene l'assegnamento ai parametri formali che vengono detti **parametri attuali.** E' quindi importante che i parametri invocati al momento della chiamata siano dello stesso numero e dello stesso tipo dei parametri formali.

Per capire quanto affermato proponiamo la dichiarazione di sottoprogrammi e relativi esempi

function <nome_sub> (lista_parametri);

Per restituire un valore si usa la parola chiave

return seguita dal nome della

variabile

10.2.1 Passaggio Per Valore

Come abbiamo visto nell'invocazione di una subroutine è possibile passare delle variabili. Quest'operazione è

suddivisa in due tipi: **per indirizzo** e **per valore.**

Supponiamo di voler scambiare due variabili utilizzando il pseudocodice scritto in precedenza.

```
<?php
function scambio($a,$b){
$temp= $a;
```

```
$a= $b;
$b= $temp;
echo "Valore var1,var2 dopo
}

$var1= 3;
$var2= 4;
echo "Valore var1,var2 dopo
scambio($var1,$var2);
echo "Valore $var1 dopo sca
echo "Valore var1,var2 dopo
?>
```

Il programma restituisce il

seguente output:

Figura: Output del programma che scambia due variabili attraverso invocazione di procedura con passaggio per valore

Come si può notare il valore della variabili nella procedura vengono scambiati ma all'esterno nulla è variato.
Questa è una proprietà del passaggio per valore poichè il valore delle variabili viene prelevato e copiato

in altre variabili (create in automatico) e su queste vengono eseguiti i vari calcoli.
E' molto utile utilizzare questo tipo di invocazione qualora si effettuino delle assegnazioni a variabili alle quale non vogliamo

cambiare il valore di output poichè come si nota dal codice appena esaminato, la variabile var1 ha valore 3, assume valore 4 nella procedura, ma alla terminazione acquisisce il suo valore originale (cioè 3).

N.B. Estremamente

sconsigliato è l'utilizzo di echo che stampano i valori di variabili all'interno di una procedura. E' opportuno farsi restituire il valore e stamparlo successivamente.

10.2.2 Passaggio Per Indirizzo

Nel passaggio per indirizzo è possibile modificare il valore delle variabili invocate nella procedura poichè,

invece di duplicare il valore ed utilizzare una variabile dummy, si usa l'indirizzo di locazione (indirizzo della memoria) della variabile originale sicchè ogni modifica viene effettuata sul dato originale.

La differenza a livello di sintassi è

l'aggiunta del
carattere & prima
della variabile.
Otteniamo cosi:

```
<?php
function  scambio(&$a,&$b)
$temp=  $a;
$a=  $b;
$b=  $temp;
echo  "Valore var1,var2 dopo
}
```

Nell'invocazione
nulla varia, quindi

riprendiamo il
codice precedente

```
$var1= 3;
$var2= 4;
echo "Valore var1,var2 dopo
scambio($var1,$var2);
echo "Valore $var1 dopo sca
echo "Valore var1,var2 dopo
scambio($var1,$var2);
?>
```

che risulta essere
in output:

Figura: Output del programma che scambia due variabili attraverso invocazione di procedura con passaggio per indirizzo

Come si nota le

variabili modificate all'interno della funzione non riassumono il loro valore originale

11. INTERAZIONE CON IL WEB

Finora, abbiamo visto costrutti che anche molti altri linguaggi di programmazione, nelle varie sfacettatture

posseggono. Esaminiamo in questo e nei successivi capitoli quelle caratteristiche che rendono unico il PHP.

11.1 Interazione browser-

server

Quando il browser effettua una richiesta al server invia delle informazioni.

Il server è in grado di accedere a diversi tipi di informazioni relative allo scambio di informazioni e alla

configurazione del server.

Esaminiamo le variabili che contengono le informazioni EGPCS (Environment, GET, POST, Cookie e Server) e altro ancora.

VARIABILE	SCOPO
$_SERVER	Conserva info

	sul server
$_ENV	Conserva info sulle variabili d'ambiente
$_GET	Conserva i valori inviati tramite GET
$_POST	Conserva i valori passati da un modulo HTML inviato tramite POST
	Conserva gli

$_COOKIE	eventuali cookie passati tramite una richiesta HTTP
$_FILES	Conserva informazioni relative ai file che il browser invia al server
$_SESSION	Conserva informazioni relative alle sessioni

	aperte
$_REQUEST	Contiene le informazioni di $_GET,$_POST e $_COOKIE

Ognuna di questa variabile rappresenta un array. Per capire quali sono tutte le coppie (key,value)

e quindi
estrapolare tutte le
info è possibile
eseguire il
seguente script.
Segue un esempio
per estrapolare
tutti i valori di
$_SERVER

```
<table  border cellpadding="
<tr><th>  Variabile </th><
<?php
foreach($_SERVER  as  $indic
```

```
echo   "<tr><td>$indice</td>
?>
```

11.2 Processare dati utente da type="text">

Nel capitolo 7 di HTML, abbiamo visto come

permettere l'inserimento di dati da parte dell'utente.E' necessario gestirli in qualche modo.

11.2.1 GET

Vediamo un esempio di form, che spedisce i dati ad una pagina php (presente nella stessa cartella)

chiamata

gestisciDatiGET.php

la cui unica

operazione è di

stampare i dati.

Il codice che segue

lo inseriamo in un

file denominato

EsempioForm.html

```
<form method="GET"  action
Inserisci tuo nome : <i
Inserisci tuo cognome
<input  type="submit"  value
```

```
</form>
```

Il file gestisciDatiGET.php è invece siffatto:

```
<?php
echo  "Welcome ".$_GET[ 'N
?>
```

Premendo il bottone invia, i dati vengono inviati e nella barra dell'indirizzo è

possibile vedere il risultato.

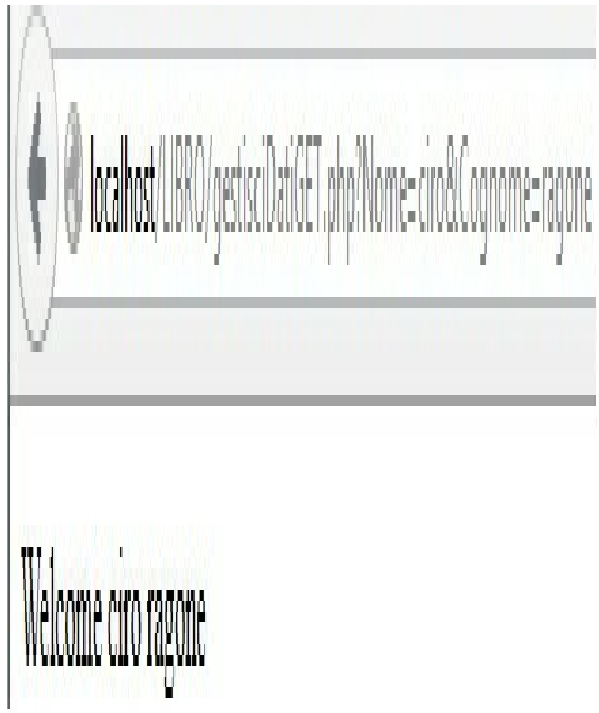


Figura 4. Risultato invocazione GET

Come si può notare, si è usato

l'array $_GET indicando che siamo interessati alla variabile Nome e Cognome identificate dall'attributo name .L'url contiene i dati inviati dall'utente (invio in chiaro) ed è per questo motivo che per i dati sensibili è

necessario usare $_POST .

Infine ogni variabile è separata dal carattere &

11.2.2 POST

Modifichiamo la pagina *EsempioForm.html* definendo come method POST e come action

gestisciDatiPOST.

Il codice che segue lo inseriamo in un file denominato *EsempioForm.html*

```
<form method = "POST" a
Inserisci tuo nome : <i
Inserisci tuo cognome
<input type="submit" value
</form>
```

Il file gestisciDatiPOST.php è invece siffatto:

```
<?php
echo  "Welcome ".$_POST[ '
?>
```

Premendo il
bottone invia, i
dati vengono
inviati e nella barra
dell'indirizzo è
possibile vedere
che il link è il
seguente:
http://localhost/gestisciDatiP
e come contenuto

viene mostrato *Welcome Ciro Ragone*

Come si può notare, si è usato l'array $_POST indicando che siamo interessati alla variabile Nome e Cognome identificate dall'attributo name .L'url NON contiene i dati

inviati dall'utente

11.3 Processare dati di Radio Button

Vogliamo ora prelevare la scelta fatta da radiobutton,

ricordando che il
name deve essere
lo stesso per tutti i
controlli dello
stesso gruppo

```
<form method="POST" acti
Scegli il condimento c
<input type="radio" name="
<input type="radio" name=
<input type="radio" name=
<input type="submit" value
<input type="reset" value="
</form>
```

Il file
SceltaCondimentoRadio.php
è siffatto:

```
<?php
echo  "Condimento scelto ".$
?>
```

Come si nota non
c'è molta
differenza con ciò
che abbiamo visto
in precedenza
poichè il valore che
abbiamo inviato è

unico

11.4 Processare dati di Checkbox

Nel caso di scelte multiple, il discorso cambia perchè il name deve avere le [] poichè verrà

inviato un array.

```
<form method="POST" acti
Scegli il condimento c
<input type="checkbox" nam
<input type="checkbox" nam
<input type="checkbox" nam
<input type="submit" value
<input type="reset" value="
</form>
```

Il file SceltaCondimentoCheck.php è siffatto:

```php
<?php
$salsa =  $_POST['condimen
echo "I condimenti che prefe
$cnt =count($salsa)
for ($i=0;$i<$cnt;$i++){
echo "Condimento ";
echo $i+1;
echo ": $salsa[$i]  <br/>";}
?>
```

Lo script è diverso da quelli visti in precedenza. Alla prima riga

recuperiamo i risultati, poi tramite il count (riga 3)recuperiamo il numero di elementi selezionati e tramite un ciclo analizziamo tutti gli elementi del vettore (riga 4-8)

11.5

Processare dati di Select

Nel caso di select bisogna distinguere il caso di scelta singola o multipla

Nel primo caso lo script php utilizzato è lo stesso dei

radiobutton,
altrimenti delle
checkbox

Mostriamo solo il
codice HTML per la
scelta singola.

Per la scelta
multipla
aggiungere le [] a
name.

```
<form  method="POST"  acti
Scegli il condimento c
<select name="condimento"
```

```
<option value="ragù">Ragù
<option value="panna">Pann
<option value="genovese"  >
</select>
<input  type="submit"value=
<input  type="reset"  value='
</form>
```

12. CONSERVAZIONE DELLO STATO

12.1 Cookies

E' spesso necessario conservare delle

informazioni anche quando la pagina web viene chiusa o si passa da una pagina all'altra.
Per memorizzare le info, poichè HTTP è stateless, analizziamo dapprima i cookies e poi le sessioni.
Un cookie è una stringa (nome=valore)

contenente dati che vengono generati dal server e conservati dalla macchina su cui gira il client. Il browser deve essere configurato per accettare i cookie i quali vengono automaticamente trasmessi indietro dal client al server.

Per settare un cookie si usa la funzione setcookie(nome, valore, expire, path, domain, secure)

Nome: Unico parametro richiesto. Serve per accedere al valore (se esiste)

Valore: Valore associato a quel

nome. Può essere una variabile, una stringa..
Expire: Indica per quanto tempo il cookie deve essere memorizzato. Se non specificato, il cookie si cancella alla chiusura del browser.
è specificato indicando il numero di secondi

dalla mezzanotte del 1/1/1970 GMT (Es. time()+60*60*2 farà scadere il cookie tra due ore)
Path:Specifica la directory sul server a cui è associato il cookie che quindi, sarà accessibile da tutte le URL residenti nel sottoalbero

radicato nel path indicato.

Domain:Indica il dominio in cui il cookie è visibile. (Es. Se "domain=amazon.it" e "path=/" il cookie è visibile anche in www.aaa.amazon.it, www.bbb.amazon.it...)

Secure: Indica che il cookie deve essere inviato solo

quando browser e server sono connessi tramite HTTPS od un altro protocollo sicuro

Per leggere il contenuto di un cookie, si fa riferimento all'array associativo

`$_COOKIE`

utizzato come visto per `GET` e `POST`.

Per memorizzare
più info in un
unico cookie si usa
la & concatenando
le varie coppie
chiave:valore.
Per cancellare un
cookie si usa
setcookie (nome,valore,0)
cosi da settare a 0
l'expire
Es.

```
<?php
```

```
setcookie('cliente', 'Ci
echo $_COOKIE['cliente'
$Utente="cliente=nome:c
setcookie('cliente', 'Ut
echo $_COOKIE['cliente'
setcookie('cliente', 'Ut
echo $_COOKIE['cliente'
?>
```

12.2

Sessioni

Per poter

propagare valori e

scelte da una
pagina Web
all'altra si usano le
sessioni
Per far partire una
sessione si usa
session_start(), per
settare un valore
$_SESSION ['Nome']=$valore
e per accedere
basta fare il
viceversa ovvero
$valore= $_SESSION ['Nome'
Per cancellare le

sessioni si usa

```
session_destroy();
```

File1:

```
<?php
session_start();
$_SESSION['count']=0;
?>
```

File2:

```
Salve visitatore, hai
<?php
echo ++$_SESSION['count
?> volte
```

13.

ACCESSO A DATABASE MySQL

13.1 Comandi MySQL

Prima di avventurarci con la

connessione del database MySQL a PHP, riepiloghiamo i maggiori comandi di MySQL

13.1.1 Creare Database

Il primo passo è creare il database a cui connetterci. Il nostro DB si chiamerà da

adesso in poi *libro*

```
create database
libro;
```

13.1.2 Creare Tabelle

Il passo successivo è la creazione delle tabelle.
Negli esempi successivi considereremo di avere una tabella

utenti con

campi:nome,cognome,eta,sti

Per creare la

tabella è

necessario

utilizzare la

keyword

Create table nome_tabella

che per il nostro

esempio è

create table utenti

(nome varchar(16),cogn

13.1.3 Select

```
select nome_campi from n
```

e volendo estrarre
nome e cognome
dalla tabella utenti
per coloro che
sono maggiorenni
scriviamo:

```
select nome,cognome from
```

Per selezionare
tutti i campi si usa

* come nome_campi. Si noti che where non è obbligatorio

13.1.4 Update

Per aggiornare informazioni del database è necessario usare

```
update nome_tabella set
```

Es. aggiornamento stipendio di 30

euro per coloro
che hanno più di
60 anni.

```
update utenti set stipen
```

Si noti che where
non è obbligatorio

13.1.5 Insert

```
insert into nome_tabella
```

Es. inserimento di
Ciro Ragone di età
23 con stipendio

1000.

```
insert into utenti (nome,
```

13.1.6 Delete

```
delete from nome_tabella
```

Es. Eliminazioni
utente chiamato
'Ciro Ragone'.

```
delete from utenti where
```

Se le condizioni
vengono omesse,

tutti i dati contenuti nella tabella sono eliminati

13.2 Dialogo con PHP

13.2.1 Connessione al DB

Per permettere la connessione PHP-MySQL è necessario avere a

portata di mano username e password per accedere a MySQL. Nel mio caso ho impostato come user e pass rispettivamente *root* e *ciro*. Il server a cui connettersi è localhost poichè lavoriamo in locale e il db è *libro*

Per connettersi la

funzione è

mysqli_connect ($server,$us

```
<?php
$host =  "localhost";
$user =  "root";
$pass =  "ciro";
$db =  "libro";
$link =mysqli_connect($host,
?>
```

13.2.2 Chisura Connessione

Per chiudere la

connessione PHP-MySQL non è necessario compiere alcuna operazione poichè automaticamente chiusa alla terminazione dello script .
Qualora lo si volesse fare si usa mysqli_close (var_conn)
Per connettersi la

funzione è

mysqli_connect($server,$user

```
<?php
mysqli_close ($link);
?>
```

13.2.3 Esecuzione Query

Ciò che vogliamo fare su un DB è interrogarlo, ovvero eseguire

delle query.
Per questo si usa
mysqli_query
(var_conn,string_query)
e i risultati
vengono
memorizzati in una
variabile per essere
esaminati.
Volendo ad
esempio eseguire
la select analizzata
nel paragrafo
13.1.3 basta

salvare la query in una stringa e passarla come parametro della funzione

Ovviamente, prima di eseguire queste operazioni è necessario connettersi al DB

```
<?php
$query="select nome,cognom
$result=mysqli_query ($link,$
```

```
?>
```

13.2.4 Esaminare i risultati

Il passo successivo è mostrare i risultati ottenuti dall'interrogazione. Come prima operazione controlliamo che una qualche

risposta sia stata ottenuta

Per questo si usa mysqli_num_rows (var_risultato) che restituisce il numero di risultati

Gestiamo i dati attraverso mysqli_fetch_assoc (var_risultato)

Quindi, dopo aver eseguito la query (che come select

ha due campi nomi
e cognome)
eseguiamo il
seguente codice:

```
<?php
if (mysqli_num_rows ($resul
    while ($row=mysqli_fetch_
        echo "Utente:".$row["co
?>
```

13.2.5 All together now

Esprimiamo le

nostre conoscenze per estrarre i dati dalla base di dati e li intabelliamo.

I passi quindi da compiere sono: connessione al DB, esecuzione della query, gestione dei dati

```php
<?php
$host =  "localhost";
$user =  "root";
```

```
$pass =  "ciro";
$db =  "libro";
$link =mysqli_connect($host,
$query="select nome,cognom
$result=mysqli_query ($link,$
if (mysqli_num_rows ($resul
    echo  "<table border=2><t
    while ($row=mysqli_fetch_
        echo "<tr><td>".$row["
echo "</table>";
```

Il risultato è il seguente

Nome	Cognome
ragone	ciro
rossi	mario
pipo	pippo

Figura 5.
Gestione dati interrogazione al DB

13.2.6 Intabellare dati a partire da

qualsiasi query

Supponiamo di avere molte queries da effettuare, ognuna con i propri campi. Se provate a farlo noterete che lo schema è sempre lo stesso, però i nomi dei campi sui quali ciclare è differente, infatti

potremmo avere più di due campi, ognuno col proprio nome.

Potremmo pensare di ciclare anzichè usando le chiavi (nome, cognome) gli indici (0,1..) ma ci sarebbe comunque la necessità di definire l'intestazione della

tabella (la parte compresa tra <thead></thead>. Avremo comunque una funzione non generale per l'esecuzione della queries.
In MySQL c'è la possibilità di recuperare il nome delle colonne attraverso la funzione di

mysqli_fetch_fields
che restituisce un
oggetto.
Noi non abbiamo
visto la
programmazione
ad oggetti in PHP
ma per sfruttare
questa funzione, ci
basta sapere che
per accedere ad un
dato si usa $obj =>
nome_proprietaà
Nel nostro caso

abbiamo la proprietà *name* che rappresenta il nome del campo
Ciò che facciamo è implementare una funzione che prende in input due parametri, la query da eseguire (rappresentata da una stringa) e la variabile ottenuta durante la

connessione al DB

Il primo passo è iniziare a costruire la tabella incapsulandola in un . Dopo aver aperto successivamente il tag <table> e il tag <thead> per dichiarare l'inizio della tabella e della intestazione, dobbiamo

recuperare il nome dei campi.

Eseguiamo quindi la query alla riga 4 facendoci restituire una variabile che chiamiamo $risultato

Recuperiamo il numero di campi richiesti dalla nostra query alla riga 5

Nelle due righe

successive (6-7) cicliamo sui nomi delle campi che utilizziamo come header della tabella

Finito il ciclo chiudiamo l'intestazione con </thead> (riga 8)

A questo punto se ci sono risultati (riga 9) li gestisco altrimenti passo a

chiudere la tabella (riga 17) e restituire i risultati (riga 18) in questo caso la tabella col solo header.

Analizziamo il caso interessante: ci sono dei dati da manipolare, allora attraverso un ciclo (riga 10) inizio a preparare la tabella aprendo

una nuova riga (riga 12) e per ogni campo (quindi ciclo - riga 13) ne recupero il valore e lo incolonno (riga 14)

Finiti i valori, quindi le colonne, chiuso la riga (riga 15).

A questo punto chiudiamo la tabella alla riga 17

e la restituiamo

alla 18

```
<?php
1. function  Lettura_Dati(
2. {
3. $tabella="<div><table id=t
4. $risultato=mysqli_query($
5. $nomiColonne=mysqli_fet
6. foreach($nomiColonne  as
7.     $tabella.="<th>"$nome-
8. $tabella.="</thead>";//Ch
9. if (mysqli_num_rows($risu
10.     while($riga  =mysqli_fe
```

```
11.     {
12.     $tabella.="<tr>";//Apro
13.     foreach($riga  as $cam
14.         $tabella.="<td>".$va
15.     $tabella.="</tr>";
16.     }
17.     $tabella.="</table></d
18. return  $tabella;
19. }
?>
```

Adesso ci basta semplicemente scrivere nella

stessa pagina:

```php
<?php
$host="localhost";
$username_db="root";
$password_db="ciro";
$database="libro";
$conn=mysqli_connect($hos
$query="select * from utenti
echo  Lettura_Dati($que
?>
```

Qualora volessimo mostrare dei nomi specifici come

intestazione della tabella è possibile utilizzare la keyword as nella query.

```
select nome as Name, cog
```

In questo modo non verranno mostrati più nome e cognome ma name e surname Abbiamo ottenuto uno schema

generale per
mostrare i dati
intabellati a partire
da una query di
estrazione
qualsiasi


CSS

1.

INTRODUZIONE

CSS è l'acronimo di Cascading Style Sheets (fogli di stile in cascata). Un foglio di stile specifica come un browser deve posizionare, formattare e visualizzare i vari elementi che

compongono una pagina web. Un foglio di stile è una collezione di regole stilistiche che definiscono il look & feel degli elementi. Uno stile può essere interno ad un file .html oppure può essere un file di testo con estensione .css .
In questo modo

CSS permette di separare lo stile dal contenuto e di definire regole generali per formattare un elemento

Il browser fonde insieme le varie definizioni incontrate esaminando in cascata le definizioni

incontrate (ecco il significato di cascading)

Il termine cascading si riferisce all'ordine di applicazione delle definizioni di stile

Hanno priorità minore quelle in un file esterno, maggiore quelle all'interno dei tag

l tag servono per specificare la struttura (semantica) di un documento e lo stile associato al tag comunica al browser come l'elemento indicato dal tag deve essere formattato, posizionato e visualizzato

1.1 Definizione Stile CSS

In passato lo stile veniva definito direttamente nel tag attraverso gli attributi che attualmente sono deprecati.

Volendo, per definire lo sfondo

della pagina di colore blu in passato si utilizzava <body BGCOLOR = "blue" riducendo la leggibilità del codice.

Per definire lo stile è possibile utilizzare tre tecniche:

All'interno del tag HTML, nel tag

HEAD e in un file esterno

1.1.1. Stile inline

Il primo modo per definire uno stile per un singolo elemento HTML, che sia esso un `div`, uno `span` o altro ancora è possibile farlo

attraverso il tag

style

Esempio di stile

per un paragrafo

rosso con testo

centrato

```
<p  style="color:red;text-alig
```

che risulta essere

Esempio di stile

per un paragrafo

rosso con testo

centrato

Per adesso non

sappiamo ancora come definire le regole CSS ma esaminiamo prima dove piazzare e poi come scrivere il codice.

1.1.2. Stile Internal

Ricorderete ad inizio libro che abbiamo parlato

del tag head con le sue funzioni principali. Adesso quindi possiamo dare un senso a quelle parole

All'interno del tag head si posiziona il tag <style type = "text/css" > o più semplicemente <style>

Il codice quindi risulta essere cosi

strutturato

```
<html>
<head>
<title>Esempio</title>
<style  type="text/css">
p  {"color:red;text-align:cent
</style
</head
```

che in questo caso
formatta tutti i tag
p di rosso con
testo centrato
Ovviamente è

possibile definire lo stile nel tag head come appena esaminato, unendo formattazione al singolo elemento come visto nel paragrafo precedente.

1.1.3. Stile Esterno

Lo stile viene definito all'interno di un altro file (con estensione .css) rispetto a dove è contenuto l' HTML

Per far ciò si inserisce all'interno del tag <head> il tag <link> usato in combinazione con i tag rel,href e type .

Quindi il codice

risulta siffatto:

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

Tramite type indichiamo se stiamo considerando un foglio di stile (.css) o uno contenente codice JavaScript (.js). Con href specifichiamo il

path ed infine

attraverso rel

indichiamo il tipo

di file

Quindi volendo

richiamare un file

chiamato style.css

scriveremo

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

1.1.4.

Ereditarietà

La regola che viene applicata dipende da dove essa è definita. Viene applicata la regola inline, se non presente quella definita in head altrimenti quella definita nel file esterno.

Supponiamo di avere il seguente

codice:

```
<html>
<head>
<title>Esempio</title>
<style  type="text/css">
p  {"color:red;text-align:cent
</style>
</head>
<body>
<p  style="color:yellow">  TE
```

che risulta essere

TESTO COLORE

GIALLO

TESTO ROSSO PER LA REGOLA DEFINITA IN <STYLE>

Se si applica uno stile a un elemento, sarà ereditato automaticamente da quelli contenuti in esso. Se non specificato

diversamente, le caratteristiche di un elemento rimangono quelle predefinite o quelle definite in uno stile ereditato Le proprietà di uno stile di un livello figlio sono ereditate dallo stile del livello genitore. Ad esempio, se

abbiamo un tag <b> e in un tag <p> al quale definiamo lo stile questo verrà ereditato anche dai figli. Detto in termini di codice:

```
<p  style="color:red">  <b>SO
```

che risulta essere

SONO DI COLORE ROSSO,GRASSETTO *ed enfatizzato*

Come si nota, non abbiamo definito <b> e di colore rosso ma essendo figli di <p> ne hanno ereditato lo stile

1.2 Scrivere Regole CSS

Una regola CSS consiste di un selettore e un

blocco di
dichiarazione

Il blocco di
dichiarazione ha
delle proprietà e
dei valori

```
selettore1 { proprietà
```

Il selettore
definisce quale
HTML formattare,
nella dichiarazione
viene definita la
proprietà e il

valore. Si inizia con la { e si chiude con la }. Ogni dichiarazione è separata da un ;, i diversi valori invece sono seperati da ,
Molte proprietà accettano come valore initial e inherit per indicare di assumere il valore iniziale e

quello già definito per l'elemento genitore, infatti successivamente non li esprimeremo come possibili valori
Volendo definire ad esempio che gli elementi `h1` siano di colore blu e allineati al centro, scriveremo

```
h1 { color : blue; tex
```

Supponiamo di voler definire lo stesso insieme di regole sia per gli h1 sia per gli h2. Basta separare i vari selettori attraverso la ,.
Si noti che l'ultima regola non ha il punto e virgola.

```
h1,h2 { color : blue;
```

Qualora si voglia commentare una regola per verificare ad esempio cosa accade nel momento in cui questa viene estromessa si usa la seguente notazione: /* commento */

Per poter validare il proprio foglio di

stile si faccia riferimento al validatore CSS

2. Tipi Di Dati

Esistono diversi tipi di dati in CSS i quali permettono di specificare lunghezze, grandezze secondo unità di misura specifiche.

2.1

<number>

Questo tipo di dato rappresenta un numero, intero o frazionario.
La parte frazionaria viene espressa in decimale attraverso l'utilizzo del punto
Possono essere preceduti da segno e ovviamente i

caratteri ammessi sono quelli compresi tra 0 e 9

2.2 <percentage>

Per rappresentare un valore in percentuale si usa questo tipo di dato.

Il valore è rappresentato da

un <number>
seguito dal
simbolo % .
Permette di
stilizzare
l'elemento
aumentandone o
riducendone il
formato secondo la
percentuale
indicata
Es. Raddoppiare la
dimensione del
testo rispetto

quella originale.

```
<p style=  "font-size:200%">T
```

che risulta essere

TESTO

RADDOPPIATO

2.3 <string>

Per rappresentare una stringa come abbiamo visto si usano gli ' oppure

".

Si ricorda che per scrivere apice in apice o doppie virgolette in doppie virgolette è necessario usare il backslash.

2.4 <url>

Ad alcune proprietà è possibile assegnare

come valore un URL.

In questo caso si usa la notazione url() e l'indirizzo può essere racchiuso tra apici singoli o doppi.

Sono permessi url relativi

2.5 <color>

Per indicare un

colore ci sono tre metodi: per parola chiave, usando i codici RGB/RGBA o HSL

Se la parola chiave non esiste la regola non avrà effetto.

Alternativa alla keyword è il codice del colore preceduto dal carattere #

L'ottimizzazione

del codice suggerisce di utilizzare il codice poichè la keyword deve essere convertita in codice.

Ad esempio il codice del rosso è "#FF0000" quindi scriveremo color:#FF0000

In RGB (Red, Green, Blue) il

rosso è rgb
(255,0,0)
Per indicare anche
l'opacità si usa
rgba(255,0,0,alpha)
dove alpha varia
da 0 a 1
specificando
l'opacità.
Se non si vuole
usare l'RGBA si può
usare l'RGB e la
keyword opacity .
In HSL (Hue,

Saturation,
Lightness -
Tonalità,
Saturazione,
Luminosità) il rosso
corrisponde a
(0,100%,50%)
La versione HSLa
permette di
specificare
l'opacità

2.6 <time>

Per indicare dimensioni temporali si usano i secondi o i millisecondi. Rispettivamente i simboli sono s e ms.

2.7 <length>

Si riferiscono a misure verticali od

orizzontali. Il formato di una lunghezza è un valore. Subito dopo il valore deve essere indicata un'unità di misura senza spazio cosi come negli altri casi.
Dopo la lunghezza di valore 0 l'unità di misura è opzionale

2.7.1. Lunghezze Relative

Il suo valore è relativo ad un'altra lunghezza o proprietà

Per quanto riguarda em è uguale alla grandezza del font in uso nell'elemento o a

quella
dell'elemento
padre

Esempio a
1.2em

Per quanto
riguarda ex è pari
all'altezza del
carattere x
minuscolo
(chiamata a volte
x-height)
1ex è circa 0.5em

per molti fonts

Esempio a 1.2ex

2.7.2. Lunghezze Assolute

px :(pixel) Unità di misura relativa alla risoluzione del dispositivo con cui è visualizzato il documento (pixel)

in (inches) : (1in =

2.54cm)

cm (centimetri) .

mm (millimetri) .

pt (punti) : (1pt = 1/72 di pollice)

pc (picas) : (1pc = 12pt)

2.8 Altri tipi

Ci sarebbe ancora molto da esaminare ma per non appesantire il

capitolo, lascio al lettore desideroso di sapere il link dove potersi documentare.

I maggiori tipi li abbiamo trattati, pochi restano e propabilmente li useremo durante il resto del libro

Per maggiori info consultare questa pagina facendo

riferimento alle keyword tra parentesi angolari come ad esempio <angle>, <timing-function>

3.

PROPRIETA'

Esistono divese famiglie di proprietà: Testo, Background e colori, Font, Liste, Contenitori (box model) e Posizionamento.

3. Proprietà

Testo

Per modificare lo stile del testo ci sono diverse proprietà

3.1. Colore

La proprietà è color ed il valore è uno indicato nel paragrafo 2.5 (colori)

Codice CSS:

```
.colore{color:red}
```

Esempio:

```
<p  class="colore"> Colore
```

che risulta

Colore Rosso

3.2.

Allineamento

La proprietà è

text-align ed il

valore può essere

uno tra

left,center,right,justify

Codice CSS:

```
.allinea{text-align:ce
```

Esempio:

```
<p  class="allinea">  Testo
```

che risulta

Testo Centrato

3.3.

Decorazione

La proprietà è

text-decoration

ed il valore può essere uno tra overline,line-through,underline per indicare rispettivamente un testo sovrasegnato, barrato e sottolineato

Codice CSS:

```
.decoraSopra{text-deco
.decoraBarrato{text-de
.decoraSotto{text-deco
```

Esempio:

```
<p  class="decoraSopra">  Te
<p  class="decoraBarrato">
<p  class="decoraSotto">  Te
```

che risultano
essere (visti tutti
insieme):

Testo Overline,

~~Testo line-through,~~

Testo underline

3.4. Trasformazione del testo

La proprietà è text-transform ed il valore può essere uno tra uppercase,lowercase,capitali per indicare rispettivamente un

testo tutto
maiuscolo,
minuscolo e con le
sole lettere iniziali
maiuscole.

Codice CSS:

```
.TuttoMaiuscolo{text-t
.TuttoMinuscolo{text-t
.LettereIniziali{text-
```

Esempio:

```
<p  class="TuttoMaiuscolo">
<p  class="TuttoMinuscolo">
```

```
<p  class="LettereIniziali">  s
```

che risultano
essere (visti tutti
insieme):
TESTO MAIUSCOLO,
testo minuscolo,
Solo Iniziali

3.5.

Indentazione

La proprietà è
text-indent ed il

valore è numerico

Codice CSS:

```
.indenta{text-indent:8
```

Esempio:

```
<p  class="indenta"> Testo
```

che risulta

Testo

Indentato

3.6. Spaziatura

tra lettere

La proprietà è

letter-spacing ed

il valore è

numerico

Codice CSS:

```
.spaziaLettera{letter-
```

Esempio:

```
<p  class="spaziaLettera">  T
```

che risulta

T e s t o

3.7. Spaziatura tra Parole

La proprietà è word-spacing ed il valore è numerico

Codice CSS:

```
.spaziaParola{word-spa
```

Esempio:

```
<p  class="spaziaParola">  Te
```

che risulta

Testo parole spaziato

3.8. Spaziatura tra Paragrafi

La proprietà è

line-height ed il

valore è numerico

Codice CSS:

```
.spaziaParagrafi{line-
```

Esempio:

```
<p class="spaziaParagrafi">
 Testo spaziato</p>
```

che risulta

Testo parole

spaziato

Testo spaziato

3.9. Ritorno a capo

La proprietà è white-space ed il valore può essere uno tra normal,pre,nowrap per indicare

rispettivamente di
andare a capo
quando necessario,
di considerare i
line-break nel
sorgente e di
ignorare il ritorno
a capo.
Il nowrap continua
oltre il margine
dello spazio
dedicato anche se
il testo
effettivamente non

viene mostrato.

Per andare a capo

usare

Codice CSS:

```
.NoACapo{white-space:n
```

Esempio:

```
<p class="NoACapo"> Test
 Il testo non va a cap
```

che risulta

Il testo non va a capo se usa

3.10. Spezzare lunghe parole

Spulciamo ora le proprietà CSS3.Non esamineremo *text-emphasis* perchè non supportato dai vari maggiori browsers,*text-*

justify perchè supportato solo da IE e *text-align-last* perchè supportato solo da IE o Firefox.

La proprietà è word-wrap ed il valore può essere uno tra normal e break-word. Quest'ultimo indica che parole non spezzabili (lunga

unica parola) sia spezzata e mostrata a capo. Per questo abbiamo la necessità di creare una piccola cornice al nostro testo attraverso le proprietà width e border

Codice CSS:

```
.Spezza{word-wrap:brea
```

```
.NonSpezzare{word-wrap
```

Esempio:

```
<p  class="Spezza"> Parola
<p  class="NonSpezzare"> Pa
```

che risulta

Parola necessaria: supercalifragilistichespiral idoso
Parola necessaria: supercalifragilistichespiralido

3.11.

Ritorno a capo di parole

La proprietà è word-break ed il valore può essere uno tra normal, break-all e keep-all. Questi ultimi due permettono rispettivamente il ritorno a capo di

due lettere se il
limite è e il
proibire il ritorno.

Codice CSS:

```
.aCapo{word-break:brea
.NonCapo{word-break:ke
```

Esempio:

```
<p  class="aCapo"> Quando
<p  class="NonCapo"> Quan
```

che risulta

Quando il limite si raggiun ge due lettere andranno a

capo
Quando il limite si raggiunge la parola andrà a capo

3.12. Stile per spazio finito

Per segnalare che c'è altro testo ma che lo spazio è si usa la proprietà

text-overflow ed

il valore può essere

uno tra clip,ellipsis

e string. Per capire

le differenze

vediamo gli

esempi.

Codice CSS:

```
.clipped{text-overflow
.ellisse{text-overflow
```

Esempio:

```
<p  class="clipped">Quando
```

```
<p  class="ellisse">Quando i
```

che risulta

Quando il limite si raggiunge

Quando il limite si raggiunge

4. Proprietà Background

E' possibile stilizzare diversi elementi del background settando un colore, un immagine o altro ancora.

4.1. Colore

La proprietà è

background-color

ed il valore è uno

indicato nel

paragrafo 2.5

(colori)

Codice CSS:

```
.colore{background-col
```

Esempio:

```
<body  class="colore">
```

4.2.

Immagine

Poichè non posso settare tale proprietà per l'intero libro, simulerò lo stile in una cornice (attraverso il tag div). Qui indicherò solo il codice necessario per ottenere il nostro scopo e non per

simulare la cornice

La proprietà è background-image ed il valore è uno indicato nel paragrafo 2.4 (url)

4.3. Ripetizione Immagine

Poichè non posso settare tale

proprietà per l'intero libro, simulerò lo stile in una cornice (attraverso il tag div). Qui indicherò solo il codice necessario per ottenere il nostro scopo e non per simulare la cornice La proprietà è background-image ed il valore

è uno indicato nel paragrafo 2.4 (url) Poichè l'immagine potrebbe non riempire per intero lo schermo, nell'eventualità che ciò accada, si setti la proprietà background-repeat ed il valore può essere uno tra repeat,repeat-x,repeat-y,no-

repeat per indicare rispettivamente di ripetere orizzontalmente e verticalmente, solo orizzontalmente, solo verticalmente ed infine di non ripetere l'immagine.

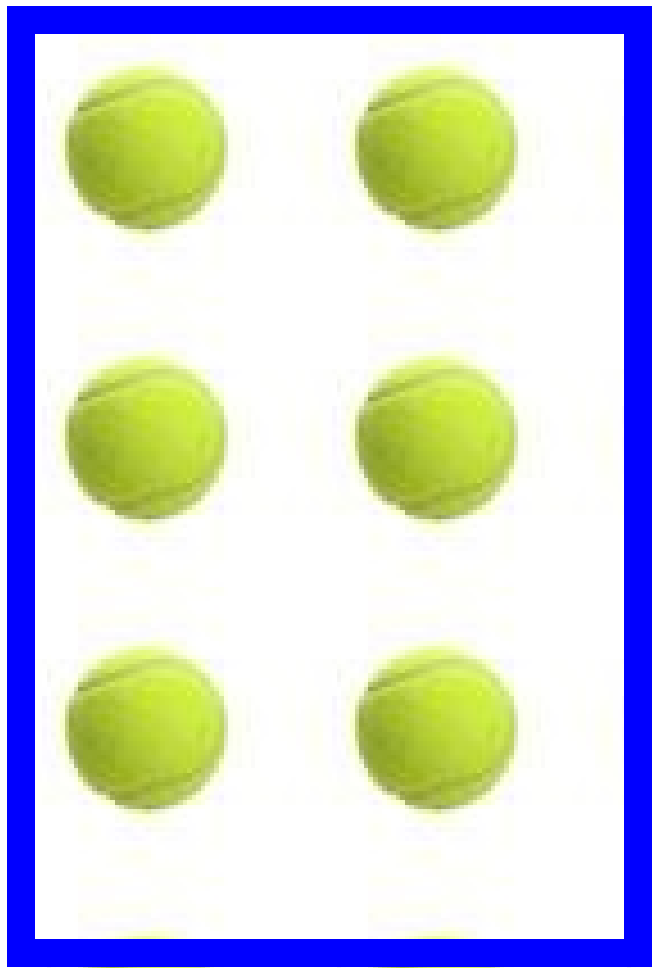
Esempio con value repeat

Codice CSS:

```
.immagine{background-i
```

Esempio:

```
<body  class="immagine">
```



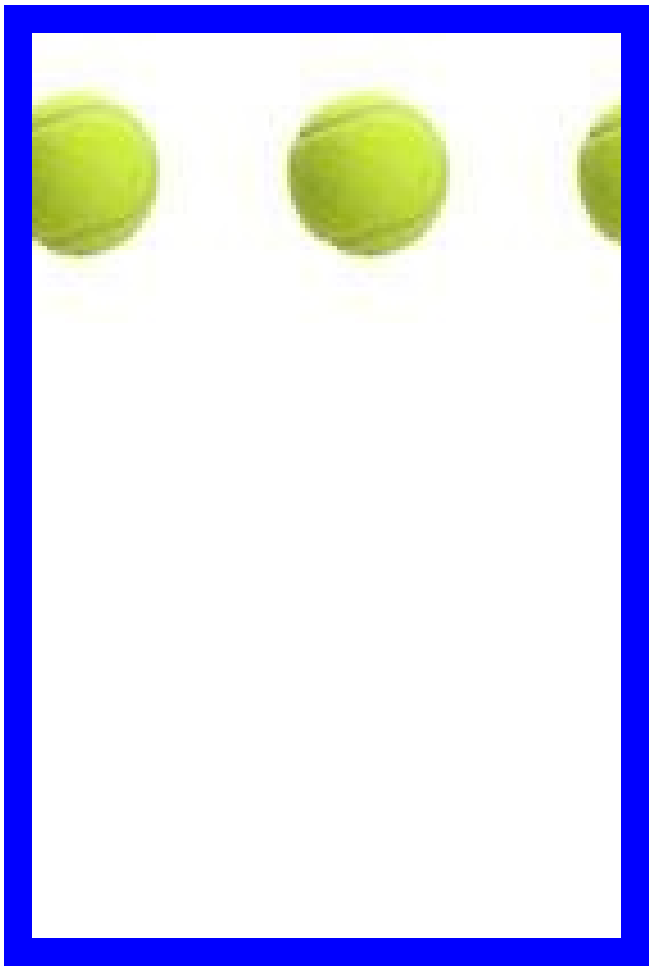
Esempio con value

repeat-x

Codice CSS:

```
.immagine{background-i
```

Il codice HTML è lo stesso di sopra

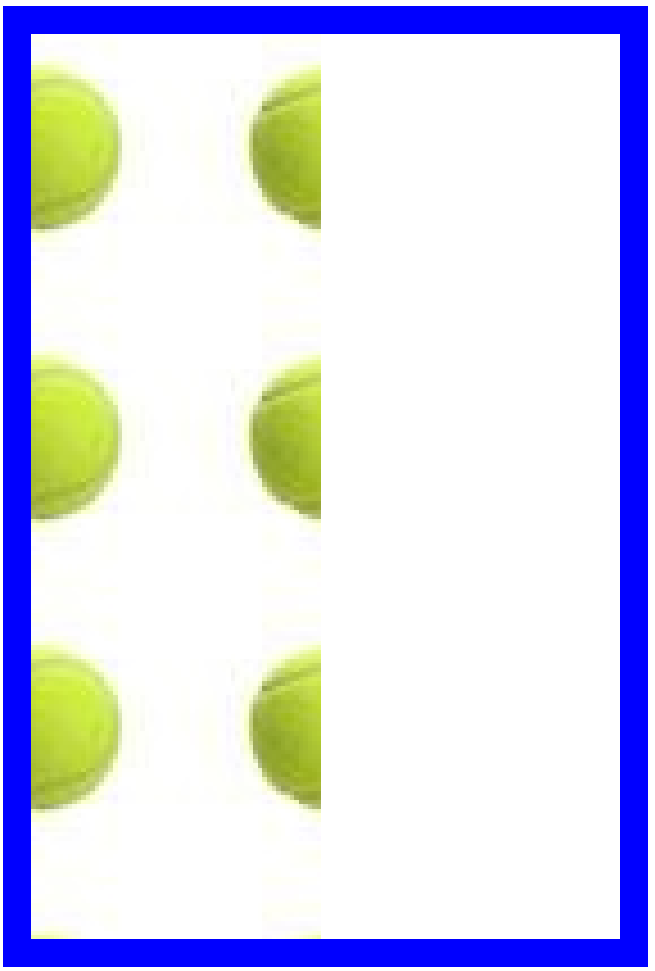


Esempio con value repeat-y

Codice CSS:

```
.immagine{background-i
```

Il codice HTML è lo stesso di sopra

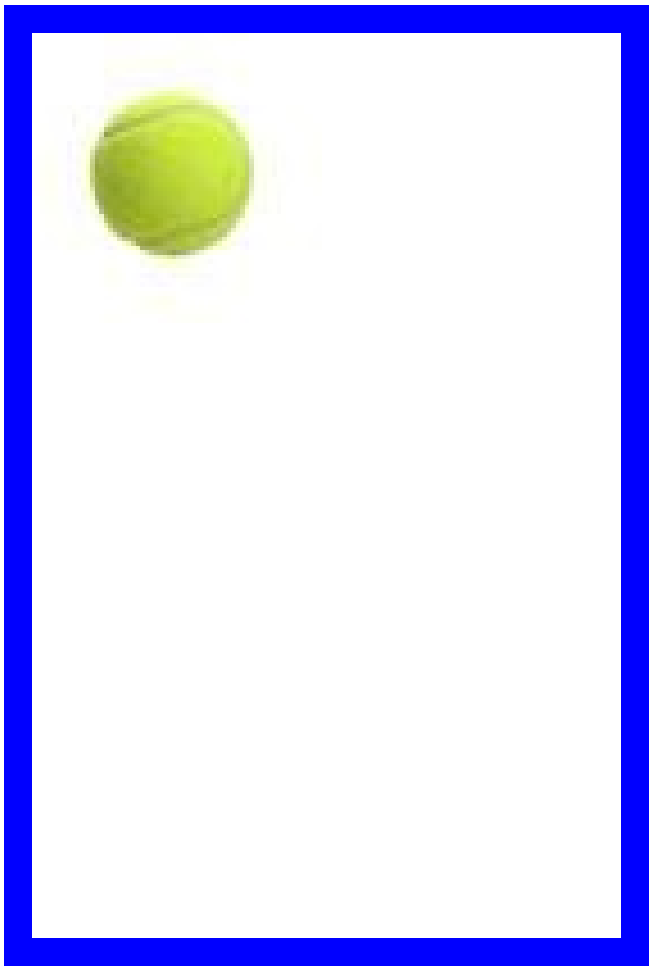


Esempio con value no-repeat

Codice CSS:

```
.immagine{background-i
```

Il codice HTML è lo stesso di sopra



4.4. Dimensione Immagine

La proprietà è background-size ed il valore è può essere uno tra auto, cover, contain, dimensione fisse o in percentuale.

auto setta

l'immagine alla dimensione originale

cover copre tutta l'area a disposizione con la possibilità che una parte non sia visibile mentre contain viene esattamente contenuta nell'area.

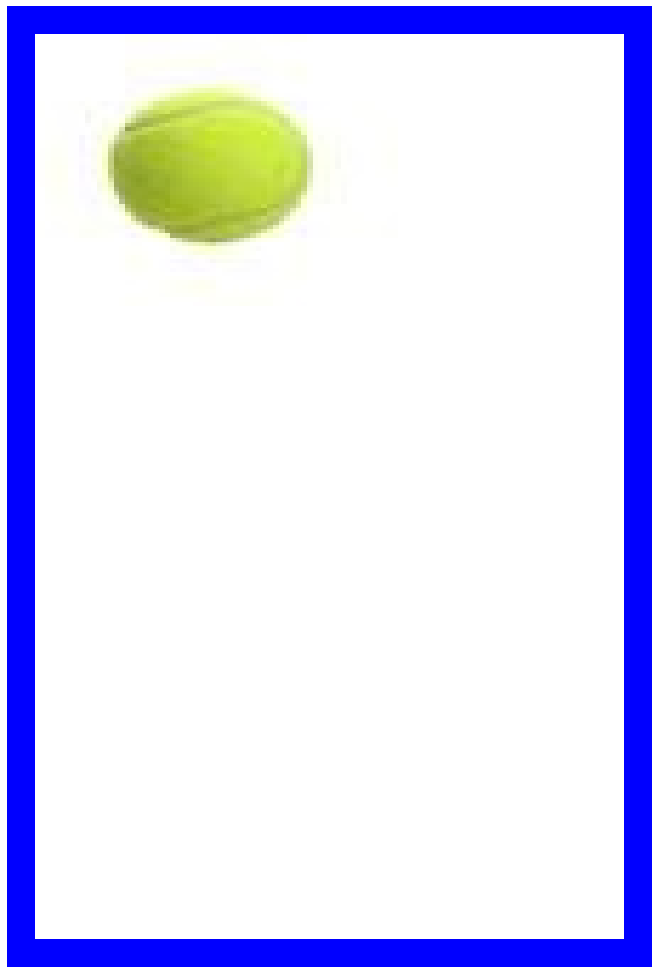
Codice CSS:

```
.dimensiona{background
```

Esempio con valori fissati:

```
<body  class="dimensiona">
```

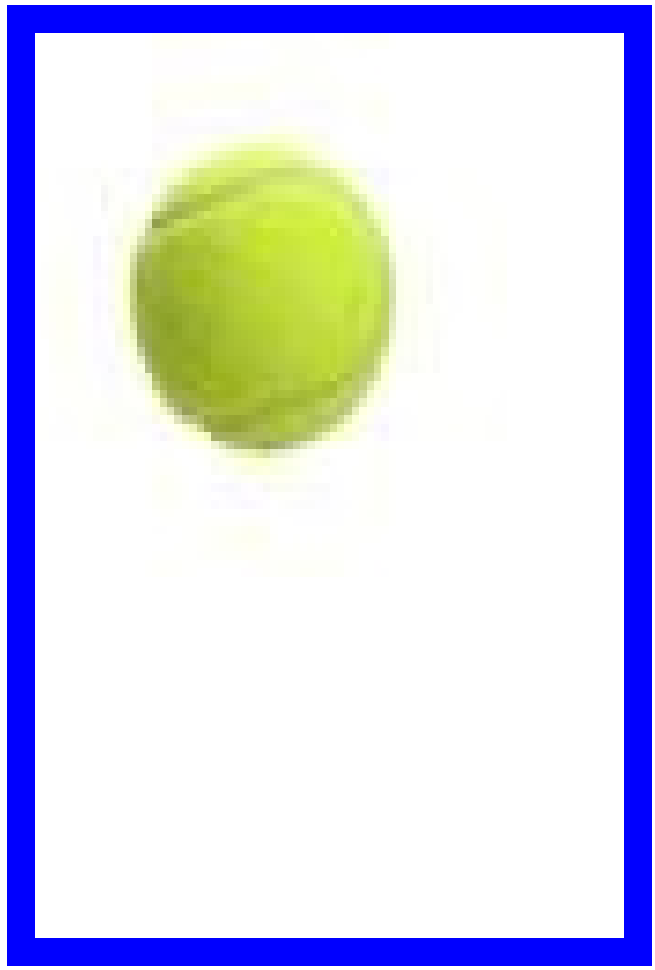
che risulta



Codice CSS:

```
.dimensiona{background
```

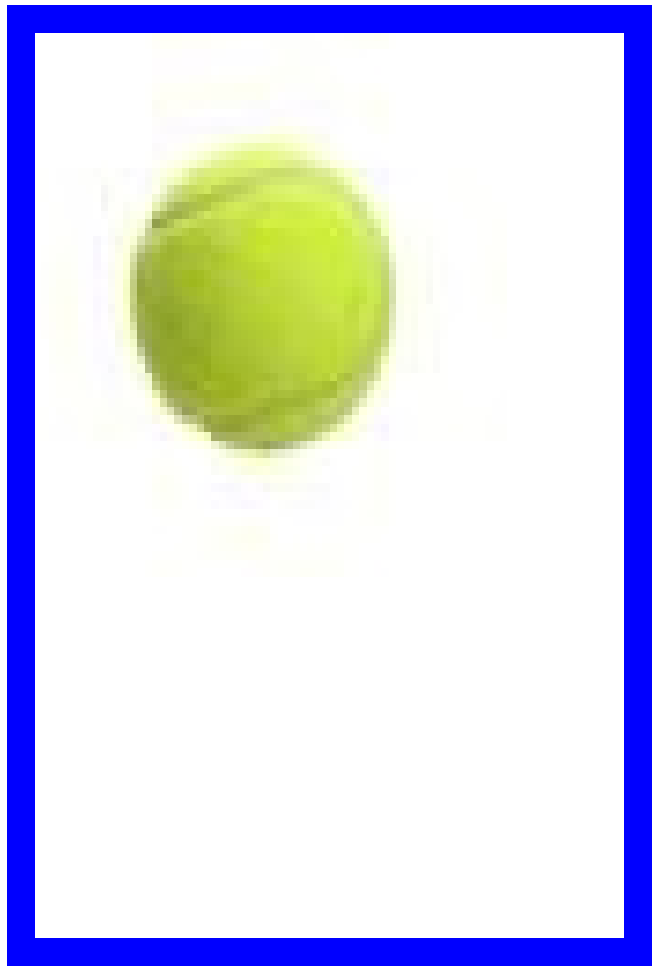
Esempio con valori in percentuale:



Codice CSS:

```
.dimensiona{background
```

Esempio con cover:

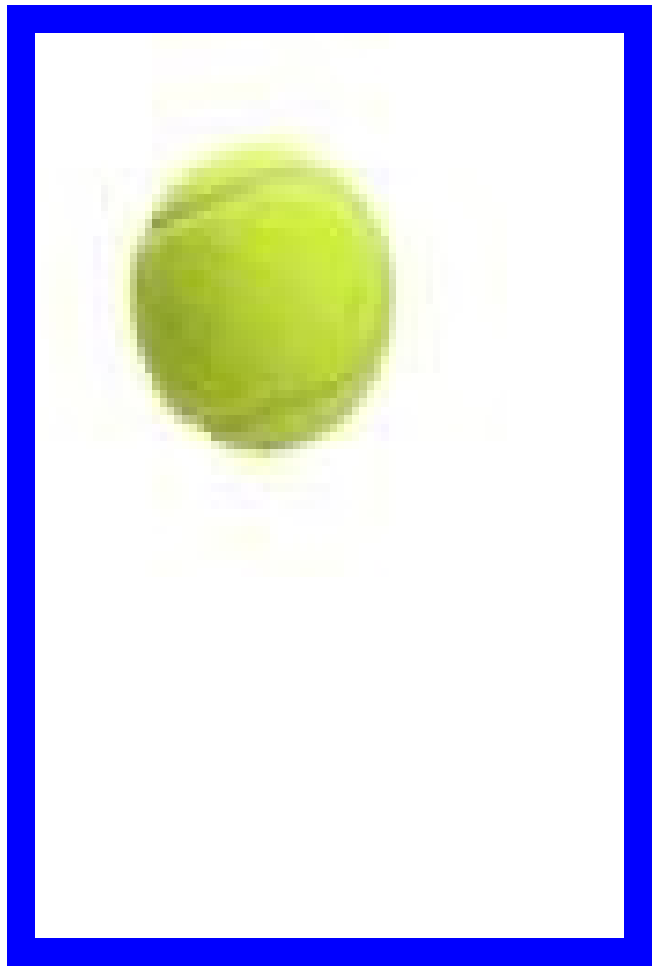


Codice CSS:

```
.dimensiona{background
```

Esempio con

contain:



4.5. Immagine In

Movimento

La proprietà è background-attachment ed il valore è può essere uno tra scroll e fixed.

Nel primo caso lo sfondo scorre lungo gli elementi, altrimenti resta fisso.

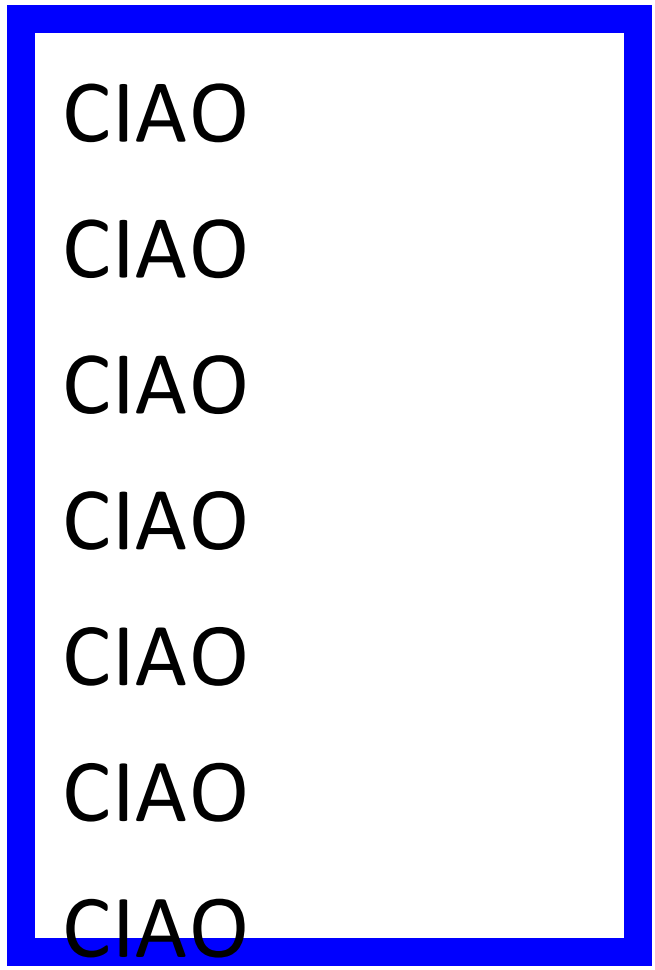
Codice CSS:

```
.fissa{background-atta
```

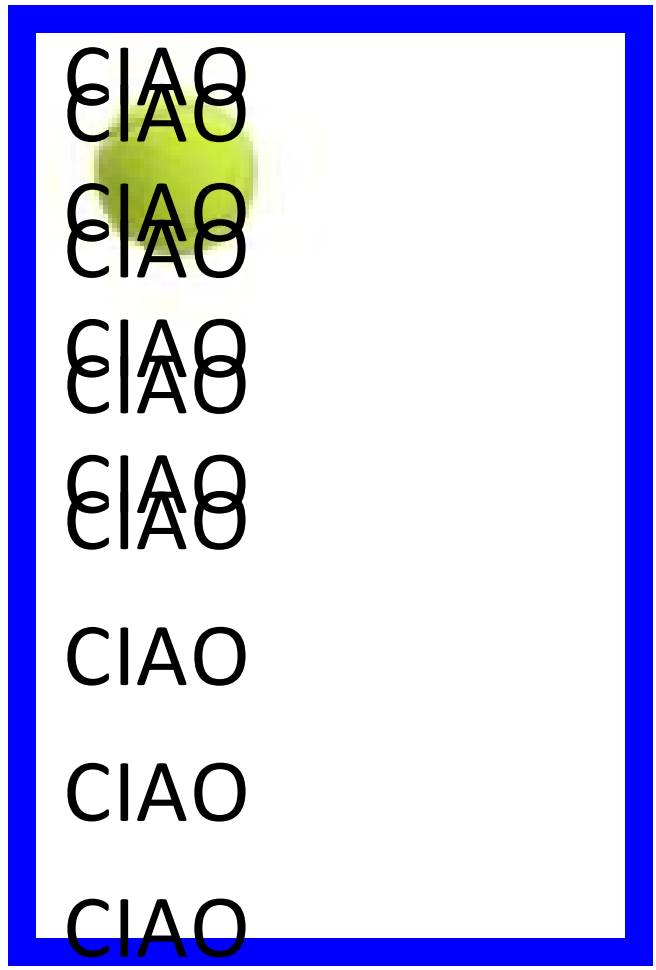
Esempio:

```
<body  class="fissa">
```

che risulta essere:



Scegliendo valore



4.6. Occupazione area

background

La proprietà è background-position ed il valore può essere numerico, in percentuale oppure uno tra {bottom,top,center} associato con uno tra {left, center, right}.

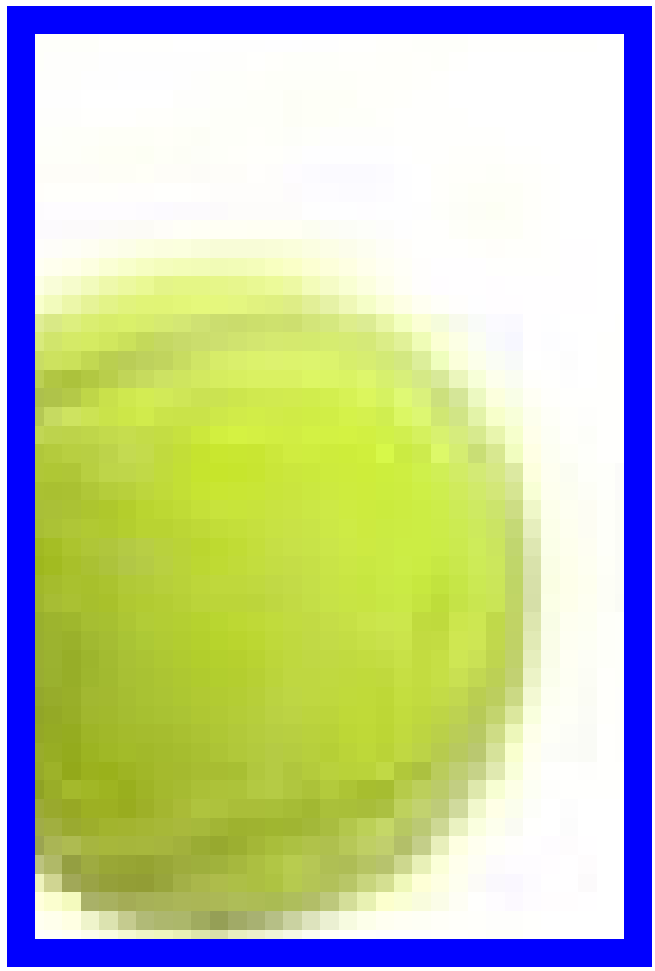
Codice CSS:

```
.posiziona{background-
```

Esempio:

```
<body  class="posiziona">
```

che risulta essere:



5. Proprietà

Font

E' possibile indicare il tipo di carattere che vogliamo usare.

5.1.

Famiglia

La proprietà è font-family ed il

valore è il nome del font.
Poichè il browser potrebbe non essere in grado di interpretare il font da noi scelto, si suggerisce di avere una lista di valori, separati dalla virgola, in modo tale che se il font non sia disponibile si prova con uno

analogo fino a procedere al nome della famiglia (quindi nome generico) alla quale il font appartiene.
Si noti inoltre che in caso il valore sia composto da più parole va circondato dalle virgolette
Codice CSS:

```
.formatta{font-family:
```

Esempio:

```
<p  class="formatta">  ESEMI
```

che risulta essere

ESEMPIO

TESTO

5.2. Stile

La proprietà è

font-style ed il

valore può essere

uno tra

normal,italic e
oblique per
indicare
rispettivamente
stile normale,
italico e obliquo.

Codice CSS:

```
.formattaI{font-style:
.formattaO{font-style:
```

Esempio:

```
<p  class="formattaI">  ESEM
<p  class="formattaO">  ESE
```

che risulta essere

ESEMPIO TESTO

ITALICO

ESEMPIO TESTO

OBLIQUO

5.3.

Spessore

La proprietà è font-weight ed il valore può essere uno tra normal,bold,bolder,lighter

o numerico per
indicare
rispettivamente
stile normale,
grassetto,
grassetto più forte,
più leggero o
decidere noi come
lo vogliamo.
Se usiamo il valore
numerico
possiamo usare
valori da 100 a 900
con passo 100

(quindi
100,200,300..)
tenendo presente
che 400
corrisponde a
normal e 700 a
bold
Codice CSS:

```
.formattaNormale{font-
.formattaBold{font-wei
.formattaBolder{font-w
.formattaLeggero{font-
.formattaValore{font-w
```

Esempio:

```
<p  class="formattaNormale"
<p  class="formattaBold">  E
<p  class="formattaBolder">
<p  class="formattaLeggero">
<p  class="formattaValore">
```

che risulta essere

ESEMPIO TESTO

NORMALE

ESEMPIO TESTO

BOLD

ESEMPIO TESTO

BOLDER

ESEMPIO TESTO

LIGHTER

ESEMPIO TESTO

600

5.4. Tutto Maiuscolo

La proprietà è font-variant ed il valore può essere uno tra normal,small-caps

per indicare rispettivamente stile normale oppure di rendere tutte le lettere maiuscole tenendo presente che le lettere maiuscole nel testo originale avranno dimensione maggiore rispetto quelle convertite.

Codice CSS:

```
.formatta{font-variant
```

Esempio:

```
<p  class="formatta">  Lette
```

che risulta essere

LETTERE MAIUSCOLE E

MINUSCOLE

5.5.

Dimensione

La proprietà è

font-size ed il

valore può essere
uno tra small,x-
small,xx-
small,smaller,medium,large,x
large,xx-
larger,larger
oppure numerico
con unità di
misura.
Codice CSS:

```
.formattaSmall{font-si
.formattaSmallx{font-s
.formattaSmallxx{font-
```

```
.formattaSmaller{font-
.formattaMedium{font-s
.formattaLarge{font-si
.formattaLargex{font-s
.formattaLargexx{font-
.formattaLarger{font-s
```

Esempio:

```
<p  class="formattaSmall">
```

che risulta essere
(per questo e gli
altri)

esempio testo a

varie dimensione: SMALL

esempio testo a varie dimensione: x-SMALL

esempio testo a varie dimensione: xx-SMALL

esempio testo a varie dimensione: SMALLER

esempio testo a varie dimensione: MEDIUM

esempio
testo a varie
dimensione:
LARGE

esempio
testo a
varie
dimensione:
x-LARGE

esempio testo a varie dimensione: xx-LARGE

esempio testo a varie

dimensione: LARGER

5.6. All In One

La proprietà per settare più valori contemporaneamente è font e vanno inseriti i valori per le seguente proprietà: font-style font-variant

font-weight font-size/line-height font-family.

font-size e font-weight sono obbligatorie, le altre se omesse assumono il valore di default.

Codice CSS:

```
.formatta{font-variant
```

Esempio:

```
<p  class="formatta"> Lette
```

che risulta essere

Lettere Maiuscole e

Minuscole

5.7. Font Personalizzato

A volte pagine web che richiedono delle elaborazioni tipografiche sofisticate hanno

bisogno di font che possono non essere disponibili sul PC dell'utente. Con CSS3 si possono specificare font anche non disponibili, fornendo un meccanismo per scaricarlo dal server, oppure per trovare tra i font

quello più simile a
quello specificato
Codice CSS:

```
@font-face{font-family
```

E' possibile usare
come proprietà
font-style e font-
weight.
Esempio:

```
@font-face{font-family
```

6. Proprietà

Liste

E' possibile modificare lo stile per settare diversi marcatori o immagini per le liste.

6.1. Immagine

Personalizzata

La proprietà è list-style-image ed il valore è il percorso dove l'immagine risiede.

Codice CSS:

```
ul{list-style-image:ur
```

Esempio:

```
<ul>
<li> Elemento </li>
<li> Elemento </li>
```

```
</ul>
```

che risulta essere (la mia immagine è una pallina da tennis)

Elemento

Elemento

6.2. Marker

Personalizzato

La proprietà è list-style-type e i valori possono essere svariati.

disc per un cerchio nero,circle per un cerchietto,square per un quadrato,decimal per i numeri,decimal-leading-zero per

numeri con 0 davanti, none per nessun simbolo. georgian, hebrew, hiragana, hiragana-iroha, katakana,katakana-iroha,armenian per la numerazione dei rispettivi Paesi. lower-alpha per i simboli dell'alfabeto tradizionale, lower-

roman per quelli romani, lower-greek per quelli greci, lower-latin per quelli latini.

Tranne per il greek esiste la versione upper (maiuscolo), quindi upper-roman,upper-alpha e upper-latin

Codice CSS (che si ripete cambiando valore):

ol{list-style-type:isc

Esempio (il codice HTML resta inviariato):

```
<ul>
<li> Elemento </li>
<li> Elemento </li>
</ul>
```

che risulta essere (per disc):

- Elemento
- Elemento

che risulta essere (per circle):

- Elemento
- Elemento

che risulta essere (per square):

- Elemento
- Elemento

che risulta essere (per decimal):

1. Elemento
2. Elemento

che risulta essere (per decimal-leading-zero):

01. Elemento
02. Elemento

che risulta essere (per georgian):

Ⴀ. Elemento
Ⴁ. Elemento

che risulta essere (per hebrew):

א. Elemento
ב. Elemento

che risulta essere (per hiragana):

あ. Elemento
い. Elemento

che risulta essere (per hiragana-iroha):

い. Elemento

ろ. Elemento

che risulta essere (per katakana):

ア. Elemento

イ. Elemento

che risulta essere (per katakana-iroha):

イ. Elemento

口. Elemento

che risulta essere
(per armenian):

Ⱡ. Elemento
Ⱡ. Elemento

che risulta essere
(per lower-alpha):

a. Elemento
b. Elemento

che risulta essere
(per lower-roman):

i. Elemento

ii. Elemento

che risulta essere
(per lower-alpha):

a. Elemento
b. Elemento

che risulta essere
(per lower-greek):

α. Elemento
β. Elemento

che risulta essere
(per lower-latin):

a. Elemento
b. Elemento

che risulta essere
(per upper-latin):

A. Elemento
B. Elemento

che risulta essere
(per upper-alpha):

A. Elemento
B. Elemento

che risulta essere

(per upper-roman):

I. Elemento
II. Elemento

che risulta essere
(per none):

Elemento
Elemento

6.3. Posizione

La proprietà è list-

style-position ed il valore può essere uno tra inside e outside.

In questo modo il marcatore viene inserito internamente od esternamente al flusso.

Codice CSS:

```
ul{list-style-position
```

Esempio:

```
<ul>
<li> Elemento </li>
<li> Elemento </li>
</ul>
```

che risulta essere
(per inside)

- Elemento
- Elemento

che risulta essere
(per outside)

- Elemento

- Elemento

7. BOX MODEL

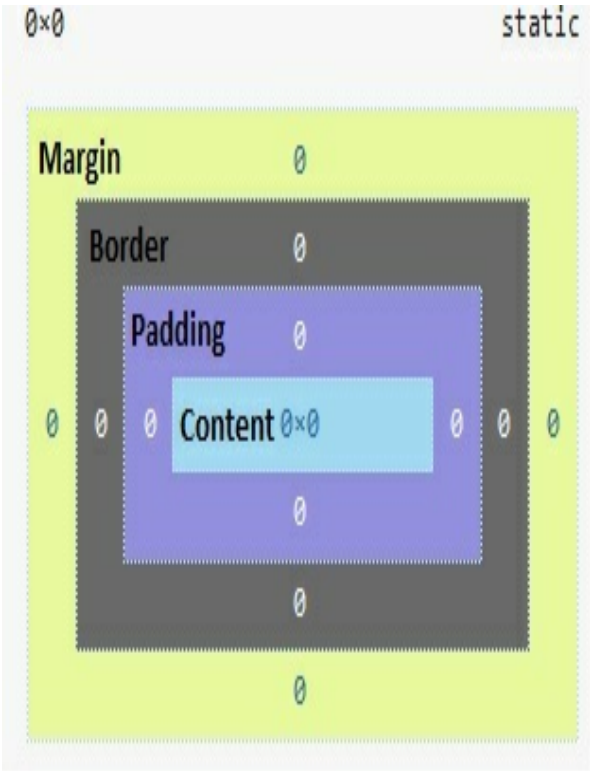


Figura: Box Model

CSS assume che ogni elemento

genera uno o più box rettangolari, chiamati elementi BOX

Tale modello è essenzialmente un rettangolo che circonda i vari elementi HTML definendo lo spazio tra di essi.

A partire dal centro:

- Content: Regione del contenuto
- Padding: Regione di respiro tra bordo contenitore e contenuto(trasparente)
- Border: Spazio dove visualizzare il bordo per il contenitore

- Margin:
  Regione che separa i vari contenitori, necessariamente trasparente.

Per questo motivo quando si progettano gli spazi nei quali inserire gli elementi si tenga presente lo spazio

occupato dai vari box e non solo dal contenuto (come potremmo erroneamente pensare).

Per calcolare la grandezza totale si usa la seguente formula:

Grandezza Totale = width + left padding + right padding + left

border + right

border + left

margin + right

margin

Per calcolare l'altezza totale si usa la seguente formula:

Altezza Totale =height + top padding + bottom padding + top border + bottom border + top

margin + bottom

margin

7.1. Proprietà Border

Permette di definire lo stile, la dimensione e il colore del bordo dell'elemento.

7.1.1. Stile

La proprietà è border-style e i

valori possono
essere svariati.
Tale proprietà
deve essere settata
affinchè le altre
sortiscano il loro
effetto
I valori possono
essere:
none|hidden|dotted|dashec
oltre ai soliti initial
e inherit.
none specifica
nessun bordo,

hidden nascosto,
dotted
punteggiato,
dashed
tratteggiato, solid
linea spessa,
double linea
doppia, i restanti
bordi 3D.
Esempio (ho
settato anche
border-width per
rendere ben
visibile la modifica

allo stile)

Codice CSS (che si ripete cambiando valore):

```
.divBorder{border-widt
```

Esempio (il codice HTML resta inviariato):

```
<div  class="divBorder">  Tes
```

che risulta essere (per none)

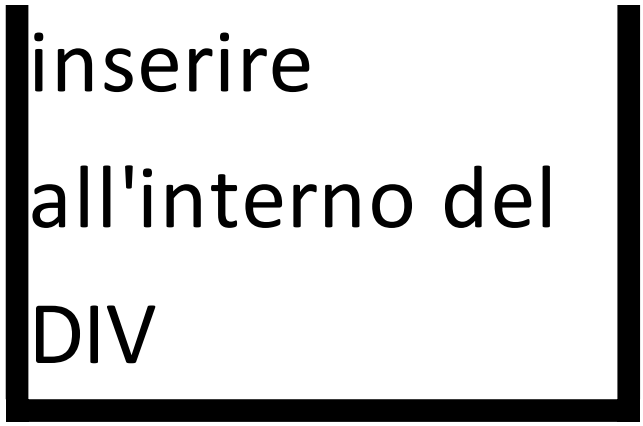
Testo da

inserire
all'interno del
DIV

che risulta essere
(per hidden)
Testo da
inserire
all'interno del
DIV

che risulta essere
(per dotted)

Testo da



che risulta essere
(per dashed)

che risulta essere
(per solid)

che risulta essere

(per double)

che risulta essere

(per groove)

che risulta essere

(per ridge)

che risulta essere

(per inset)

che risulta essere

(per outset)

In questo modo abbiamo settato lo stile per tutti i quattro lati.
E' anche possibile settare in diversi modi il bordo definendolo per ogni lato.
Per far ciò esiste la proprietà border-{left,right,top,bottom}-style

Codice CSS (che si ripete cambiando valore):

```
.divUnLato{border-widt
```

Esempio :

```
<div  class="divUnLato">  Te
```

che risulta essere

Testo da inserire all'interno del DIV che ha un solo lato con

stile

E' anche possibile stilizzare direttamente con border-style i vari lati poichè definendo i valori in senso orario (sopra,destro, sotto, sinistra).

```
.divBorderAllInOne{bor
```

Esempio :

```
<div  class="divBorderAllInOr
```

che risulta essere

Testo da inserire all'interno del DIV con i vari lati stilizzati (sopra dotted, destro dashed, sotto solid, sinistro double)

7.1.2. Dimensione

La proprietà è border-width e i valori possono essere un valore fissato, medium, thin, thick.

Gli ultimi tre permettono di avere un bordo di spessore medio, sottile e spesso.

Il valore fissato deve essere di tipo <length> , cioè numero con unità di misura.

Codice CSS (che si ripete cambiando valore):

```
.SpessoreBordo{border-
```

Esempio (il codice HTML resta inviariato):

```
<div  class="SpessoreBordo">
```

che risulta essere (per valore medium)

Testo da inserire all'interno del DIV

che risulta essere (per valore thin)

Testo da inserire

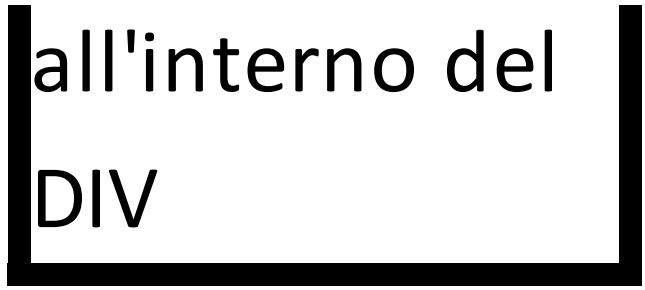
all'interno del DIV

che risulta essere (per valore thick)

Testo da inserire all'interno del DIV

che risulta essere (per valore fissato)

Testo da inserire



Come già detto per la proprietà precedente possiamo anche settare lo stile per ogni lato.
Per far ciò esiste la proprietà border-{left,right,top,bottom}-width
Come prima, è

anche possibile stilizzare direttamente con border-width i vari lati definendo i valori in senso orario (sopra,destro, sotto, sinistra).

7.1.3. Colore

La proprietà è border-color e i valori possono

essere trasparent o

<color> .

Codice CSS:

```
.ColoreBordo{border-co
```

Esempio:

```
<div  class="ColoreBordo">T
```

che risulta essere

(per valore

transparent)

Testo da

inserire

all'interno del

DIV

che risulta essere (per valore #00F ovvero blue)

Come già detto per la proprietà precedente possiamo anche

settare lo stile per ogni lato.

Per far ciò esiste la proprietà border-{left,right,top,bottom}-color

Come prima, è anche possibile stilizzare direttamente con border-color i vari lati definendo i valori in senso orario

(sopra,destro,
sotto, sinistra).

7.1.4. Outline

Per stilizzare la parte esterna del rettangolo è possibile utilizzare outline , in particolare con le proprietà outline-style , outline-width e outline-color che segueno

le stesse regole
viste per border.
Per questo motivo
illustro
direttamente un
esempio con tutte
le proprietà.
Codice CSS:

```
.OutLine{outline-color
```

Esempio:

```
<div  class="OutLine"> Test
```

che risulta essere

Testo da inserire all'interno del DIV

7.2

Proprietà

Margin

Il margine pulisce un'area attorno l'elemento (al di fuori del bordo). Il margine è completamente trasparente.

I valori possibili sono

auto, <length> o

<percentage>

Codice CSS:

```
.Respira{background-co
```

Esempio:

```
<p  class="Respira">Testo c
```

che risulta essere

Testo

da
inserire
all'interno
del
paragrafo

Come analizzato
precedentemente
è possibile gestire
indipendentemente
i vari lati del

rettangolo
attraverso margin-
{left,top,bottom,rigth}
o settandoli con
margin con la
regola del senso
orario

7.3

Proprietà

Padding

Il padding pulisce un'area attorno il contenuto (all'interno del border). Il margine è completamente trasparente.

I valori possibili sono <length> o

<percentage>

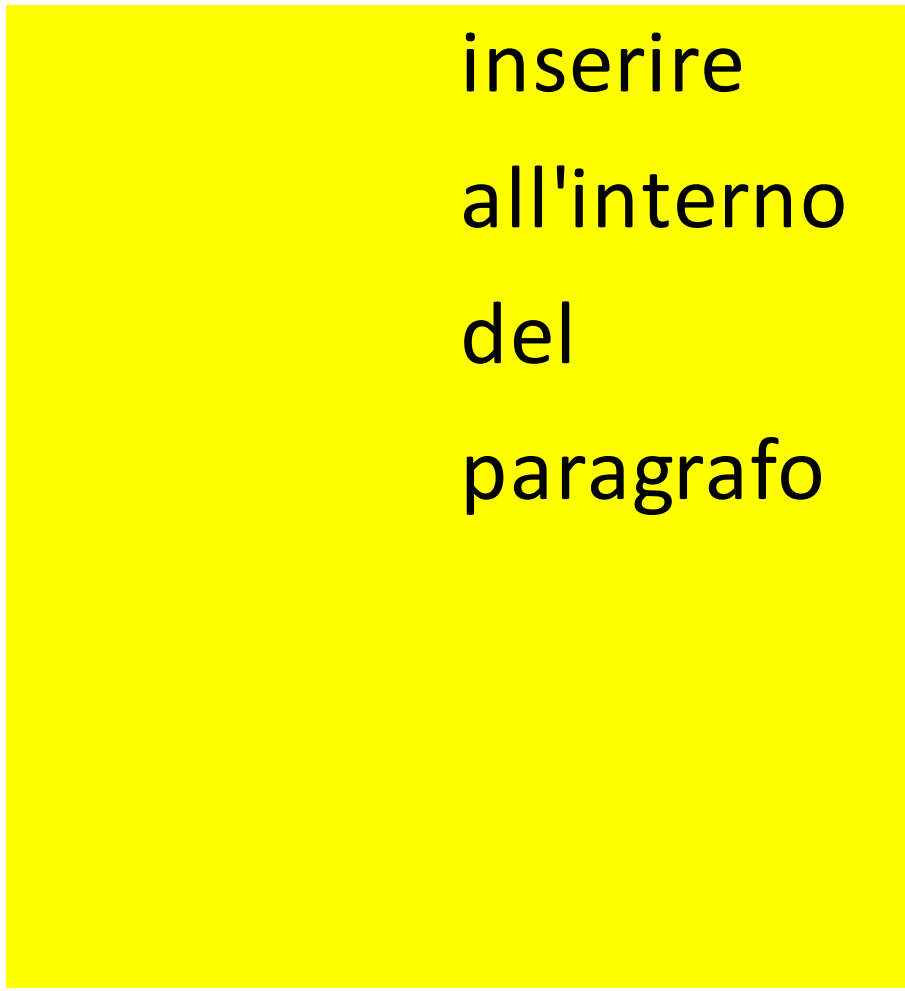
Codice CSS:

```
.Respira{background-co
```

Esempio:

```
<p  class="Respira">Testo c
```

che risulta essere



Come analizzato
precedentemente
è possibile gestire
indipendentemente
i vari lati del
rettangolo

attraverso margin-{left,top,bottom,rigth} o settandoli con margin con la regola del senso orario

7.4

Proprietà

Content

Possiamo manipolare la dimensione dell'elemento con diverse proprietà

7.4.1.

Larghezza e

Altezza

Le proprietà sono rispettivamente width e height.

I valori possibili sono auto, <length> o <percentage>

Codice CSS:

```
.Dimensiona{background
```

Esempio:

```
<p class="Respira">Testo c
```

che risulta essere

Testo da inserire all'interno
paragrafo

7.4.2.

Larghezza e Altezza Minima e Massima

In questo modo si definisce l'espansione minima o massima di un elemento.

Le proprietà sono rispettivamente max-width , min-width e max-height e min-

height .

I valori possibili

sono auto,

<length> o

<percentage>

Codice CSS:

```
.Dimensiona{background
```

Esempio:

```
<p class="Respira">Testo d
```

che risulta essere

Testo da inserire all'interno
paragrafo

8. LAYOUT

La visualizzazione di un documento con CSS avviene identificando lo spazio di visualizzazione di ciascun elemento del documento. Abbiamo visto che ogni elemento è definito da un contenitore (box)

all'interno del quale vi è il contenuto.
l contenitori sono in relazione tra loro poichè i contenitori degli elementi contenuti (figli) sono dentro al contenitore dell'elemento (genitore).

8.1 Flusso

8.1.1 Flusso normale di tipo block

I contenitori sono posti l'uno sopra l'altro in successione verticale (come i paragrafi) occupando tutta la width disponibile

In HTML gli elementi block-

level hanno un flusso normale di tipo blocco

8.1.2 Flusso normale di tipo inline

I contenitori sono posti l'uno accanto all'altro in successione orizzontale (come parole della stessa

riga) occupando solo la width necessaria

In HTML gli elementi text-level hanno un flusso normale di tipo inline

8.1.3 Flusso di tipo float

I contenitori (che sono all'interno del

contenitore
genitore) sono
spostati
all'estrema sinistra
o destra del
contenitore
genitore, lasciando
che gli altri
contenitori "vi
girino intorno"
In HTML il tag IMG
ha un flusso di tipo
float

8.1.3.1. Float

La proprietà è float ed il valore può essere uno tra right,left e none.

Con float l'elemento diviene mobile e lo si rimuove dal flusso normale

Si cambia la disposizione dell'elemento

rispetto al flusso normale, l'elemento è posizionato verso il bordo dell'elemento contenitore (a destra o a sinistra) Se si posizionano più elementi flottanti uno dietro l'altro, loro flotteranno uno di fianco all'altro.

Codice CSS:

```
img{float: right;margi
```

Esempio:

```
<p> <img src="ball.jpg"/>Te
```

che risulta essere:

Lorem Ipsum è un testo segnaposto utilizzato nel settore della tipografia e della stampa. Lorem Ipsum è considerato il testo segnaposto standard sin dal sedicesimo secolo, quando un anonimo tipografo prese una cassetta

di caratteri e li assemblò per preparare un testo campione.

8.1.3.2. Clear

E' l'opposto di float, specifica quale lato di un elemento deve essere libero da altri elementi flottanti (cioè con

la proprietà float settata)

La proprietà è clear ed il valore può essere uno tra right,left,both e none i quali rispettivamente fanno si che il lato destro, sinistro, entrambi, nessuno dell'elemento sia libero da elementi flottanti

Codice CSS:

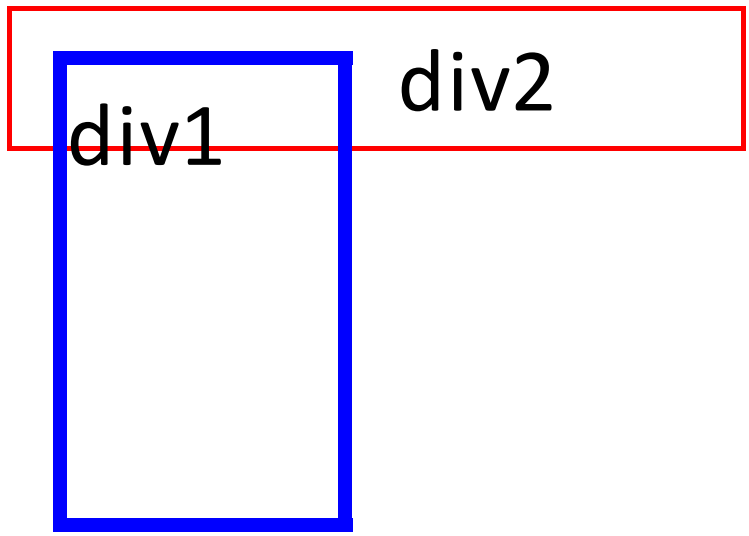
```
.div1 {float: left;wid
.div2 {border: 1px sol
h1{clear:both}
```

Esempio:

```
<h1> No Clear </h1>
<div class="div1" div1 </d
<div class="div2" div2 </d
```

che risulta essere
(con clear):

Clear



TESTO

Eliminando la

regola per `h1`

Codice CSS:

```
.div1 {float: left;wid
.div2 {border: 1px sol
```

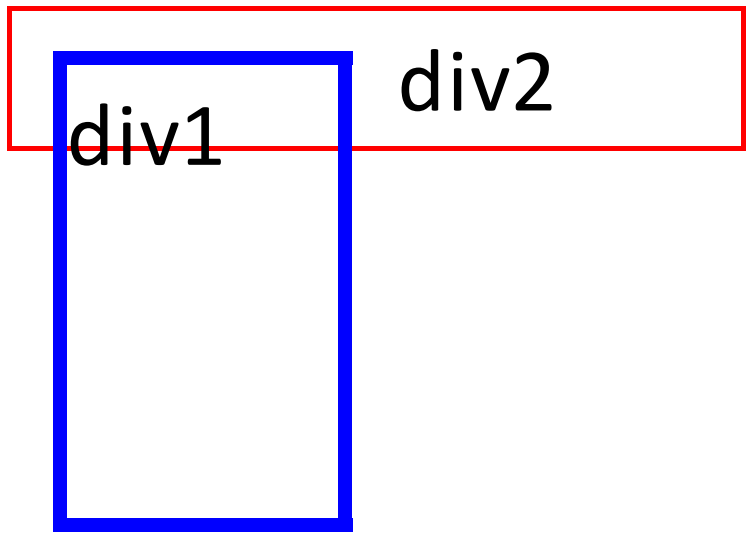
Esempio:

```
<h1> No Clear </h1>
<div class="div1" div1 </d
<div class="div2" div2 </d
```

che risulta essere

(senza clear):

No Clear



TESTO

8.1.4 Posizionamento

I contenitori vengono posti nella posizione indicata dal valore della proprietà position

9. ID e

CLASS

Come già detto la maggior parte dei tags può avere l'attributo ID o CLASS

Personalmente, uso ampiamente ID quando devo identificare gli elementi per poi manipolarli in JavaScript mentre

uso class in CSS per assegnare lo stile a più elementi di uno stesso tipo

9.1 CLASS

CLASS :assume un valore stringa qualunque. Più elementi possono condividere lo stesso valore. Questo permette

di assegnare gli elementi ad una classe

La sintassi per la definizione nel tag style è

```
nome_tag.nome_classe {pro
.nome_classe {proprietà:v
```

La sintassi nel file HTML è

```
<tag class="nome_classe">
```

Esempio, codice

nello <style> :

```
.Didascalia{color:red}
```

Nell'HTML

```
<p class="Didascalia"> TES
```

produce cosi in output:

TESTO DIDASCALIA,
VIENE MOSTRATO
DI ROSSO

9.2 ID

La sintassi per la definizione nel tag style è

```
nome_tag#nome_ID {propr
#nome_classe {proprietà:
```

La sintassi nel file HTML è

```
<tag id="nome_ID">
```

10.

PSEUDOCLASSI

Alcune caratteristiche della visualizzazione di un documento non sono direttamente dipendenti dalla struttura del documento ma sono piuttosto funzioni

dell'interazione utente-browser

Es. L'utente passa col mouse sopra ad un link

La sintassi in generale è selettore:pseudoclasse{regole

10.1 Links

Quando passiamo su un link, quando un link non è

ancora visitato, quando è già stato visitato

La pseudoclasse :link si applica ai link che non sono stati ancora visitati

La pseudoclasse :visited si applica quando un link è stato visitato dall'utente

La pseudoclasse :hover si applica

quando l'utente ha selezionato un elemento, ma non lo ha attivato (è sopra il link con il cursore del mouse)
La pseudoclasse :active si applica quando un elemento è stato attivato dall'utente (dal momento che l'utente preme il bottone del mouse

al momento in cui
lo rilascia)

Codice CSS:

```
a:link {color:red}/* lin
a:visited {color:blue}
a:hover {color:yellow}
a:active {color:lime}/
```

Esempio:

```
<a> Link </a>
```

che risulta essere:

Link.

Disporre le definizioni in questo ordine, altrimenti le regole a cascata possono non fare attivare alcune delle pseudoclassi

10.2 Classi CSS e Pseudo-

Classi

E' ovviamente possibile combinare quanto appreso finora con le nuove nozioni, ovvero combiniamo classi e sottoclassi

Supponiamo di voler stilizzare solo quelle ancore che hanno un attributo

class = "esempioAncora" .

```
a.esempioAncora:link {
```

Esempio:

```
<a  class="esempioAncora">
```

che risulta essere:

Link.

10.3 Pseudo-

Classi

Come ricordete ci sono diverse proprietà che gli elementi per l'input come <input> , <select> , <textarea> , ecc. posseggono.

A tal proposito, possiamo settare lo stile per quegli

elementi che hanno un certo stato

10.3.1 :checked

Voglio dare uno stile diverso a quegli elementi checked cioè selezionati

Valido per <input> e

<option> .

```
input:checked{width:30
```

L'HTML
(riprendendo
quello esaminato
per i radio button)
è il seguente

```
<input type="radio" name=
<input type="radio" name=
```

10.3.2 :disabled, :enabled

Servono per dare rispettivamente uno stile diverso a

quegli elementi
che hanno
l'attributo
disabled settato o
che sono attivi.

```
input:disabled{backgro
input:enabled{backgrou
```

L'HTML
(riprendendo
quello esaminato
per i radio button)
è il seguente

```
Disabilitata:<input type
Abilitata:<input type="t
```

10.3.3 :valid, :invalid

Servono per dare rispettivamente

uno stile diverso a
quegli elementi
che rispettano o
meno un criterio.

```
input:valid{background
input:invalid{backgrou
```

L'HTML è il
seguente

```
Valida:<input type="email
Invalida:<input type="em
```

10.3.4 :required, :optional

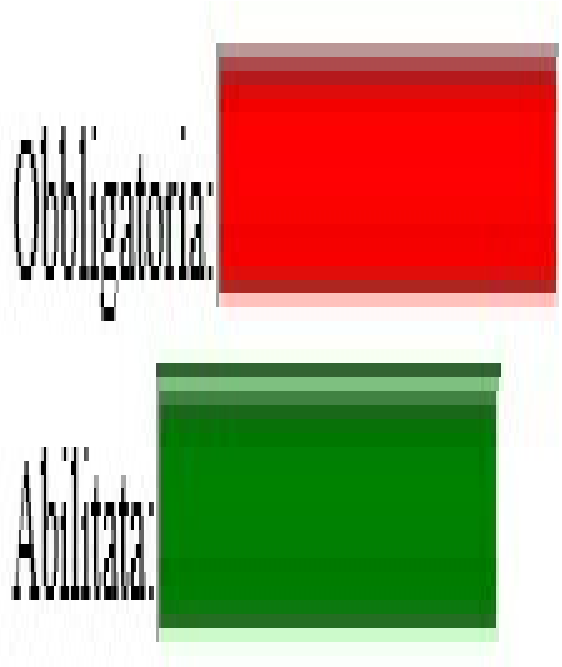
Servono per dare rispettivamente uno stile diverso a

quegli elementi
che sono
obbligatori o
meno.

```
input:required{backgro
input:optional{backgro
```

L'HTML è il
seguente

```
Richiesta:<input type="t
Non Obbligatorio<input
```



10.3.5 :in-range, :out-of-range

Servono per dare rispettivamente uno stile diverso

nel momento in cui
il valore immesso è
all'interno di un
intervallo di valori
o al di fuori.
Non valido in
Internet Explorer

```
input:in-range{backgro
input:out-of-range{bac
```

L'HTML è il
seguente

```
In Range:<input type="nu
```

Out of Range<input type=

10.3.6 :focus

Nel momento in cui il cursore è all'interno della casella di testo lo stile desiderato ha

effetto.

Valido in Internet Explorer8 solo se c'è <!DOCTYPE html>

```
input:focus{background
```

L'HTML è il seguente

```
Focus:<input type="text"/>
No Focus<input type="text
```

10.3.7 :empty

Quando il selettore non ha nessun elemento al suo interno lo stile desiderato ha effetto.

```
p:empty{background-col
```

L'HTML è il

seguente

```
Senza Elemento:<p> </p>
Con Elemento:<p> Testo
```

che risulta essere:

Testo o tags

10.3.8 :first-child

L'effetto

desiderato prende vita solo se il selettore indicato è il primo figlio di qualche altro elemento.

```
p:first-child{backgrou
```

L'HTML è il seguente

```
<div>
<p> Testo...</p>

 </div>
```

che risulta essere:

Testo...

```
<div> Testo
<p> Testo...</p>
</div>
```

che risulta essere:

Testo

Testo...

```
<div>
<h1> Primo figlio di d
<p> Testo...</p>
</div>
```

che risulta essere:

Primo figlio di div è h1

Testo...

10.3.9 :first-of-type

Con questa proprietà possiamo selezionare il

primo elemento indicato a prescindere dalla posizione occupata nell'albero.

Riprendendo l'esempio precedente (il terzo), se volessi selezionare il p che non è primo figlio del div possiamo usare questa proprietà

```
p:first-of-type{backgr
```

L'HTML è il seguente

```
<div>
<p> Testo...</p>
</div>
```

che risulta essere :

Testo...

```
<div> Testo
<p> Testo...</p>
<p> Non Selezionato</p>
</div>
```

che risulta essere:

Testo

Testo...

Non Selezionato

```
<div>
<h1> Primo figlio di d
<p> Testo comunque sel
</div>
```

che risulta essere:

Primo figlio di div è h1

Testo comunque selezionato poichè primo figlio p

10.3.10 :last-child e :last-of-type

Le proprietà complementari delle due appena viste sono rispettivamente :last-child e :last-of-type

La prima proprietà permette di selezionare solo se l'elemento indicato è l'ultimo figlio

La seconda seleziona l'ultimo

elemento di quel tipo a prescindere dagli altri tags

10.3.11 :only-child

Con questa proprietà possiamo dare stile ad un elemento se questo è l'unico figlio.

```
p:only-child{backgroun
```

L'HTML è il seguente

```
<div>
<p> Sono figlio unico.<
</div>
```

che risulta essere :

Sono figlio unico

```
<div>
<p> Ho un fratello, ch
<span> Il Fratello</spa
</div>
```

che risulta essere:

Ho un fratello, che sia p, span o altro non mi interessa.
Il Fratello

10.3.12 :only-type

Con questa proprietà possiamo dare stile ad un elemento se questo è l'unico figlio di quel tipo.

```
p:only-type{background
```

```
<div>
<p> Ho un solo fratell
<span> Il Fratello</spa
</div>
```

che risulta essere:

Ho un fratello, che è span quindi applica lo stile.

Il Fratello

```
<div>
```

```
<p> Ho un fratello, ch
<p> Il Fratello</span>
</div>
```

che risulta essere:

Ho un fratello, che
è p quindi non
sono l'unico di
quel tipo.
Il Fratello

10.3.13 :nth-child(n)

Con questa

proprietà
possiamo dare stile
all'n-esimo
elemento figlio di
qualche altro
elemento.

```
p:nth-child(1){backgro
```

```
<div>
<p> Il primo figlio è u
<span>span.</span>
</div>
```

che risulta essere:

Il primo figlio è un p.

span

```
<div>
<span> Primo span.</spa
<p> Primo p, che non v
</div>
```

che risulta essere:

Primo span.

Primo p, che non viene selezionato poichè non è il primo p figlio

10.3.14 :nth-last-child(n)

Con questa proprietà possiamo dare stile all'n-esimo elemento figlio di qualche altro elemento, contando dall' ultimo al primo.

```
p:nth-child(1){backgro
```

```
<div>
<p> Il primo figlio è u
<span>span.</span>
</div>
```

che risulta essere:

Il primo figlio è un

p.

span

```
<div>
<span> Primo span.</spa
<p> Primo p, che viene
</div>
```

che risulta essere:

Primo span.

Primo p, che viene selezionato poichè è il primo p figlio a partire da sotto

10.3.15 :nth-of-type(n)

Con questa proprietà

possiamo dare stile all'n-esimo elemento figlio di qualche altro elemento, tenendo presente il tipo.

```
p:nth-of-type(1){backg
```

```
<div>
<span>span.</span>
<p> Il primo figlio di
</div>
```

che risulta essere:

span

Il primo figlio è un p.

```
<div>
<span> Primo span.</spa
<p> Primo p</p>
<p> Secondo p</p>
</div>
```

che risulta essere:

Primo span.

Primo p

Secondo p

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

Con questa proprietà possiamo dare stile all'n-esimo elemento figlio di qualche altro elemento, tenendo presente il tipo, contando dall' ultimo al primo.

```
p:nth-last-of-type(1){
```

```
<div>
<p> Il primo figlio è u
<span>span.</span>
</div>
```

che risulta essere:

Il primo figlio è un p.

span

```
<div>
<p> Primo p.</p>
<p> Secondo p, che vie
</div>
```

che risulta essere:

Primo p.

Primo p, che viene selezionato poichè è il primo p figlio a partire da sotto

10.3.17 :not(selettore)

Qualora si voglia definire lo stile per

tutti gli elementi
ad eccezione di
qualcuno si usa il
:not.

```
:not(p){background-col
```

```
<div>
<p> Il primo figlio è u
<span>span.</span>
</div>
```

che risulta essere:
Il primo figlio è un
p.

span

10.3.18

:lang(att)

Un elemento può avere l'attributo lang per indicare il tipo di lingua.

```
p:lang(it){background-

<p> p senza lang.</p>
<p lang="it" Testo con a
```

che risulta essere:

p senza lang.

Testo con att lang

11.

PSEUDO-ELEMENTI

Servono per creare astrazioni sull'albero del documento oltre a quelle strutturali proprie del linguaggio HTML.

Si può assegnare uno stile ad un

contenuto che non esiste nel documento sorgente o far riferimento alla prima linea o lettera del contenuto di un elemento

La sintassi in generale è selettore::pseudoelemento{re

11.1 Stile

prima linea

E' possibile
assegnare uno stile
alla prima linea di
testo del tag di
ogni elemento
indicato

```
p::first-line{color:re
```

```
<p>Lorem Ipsum è un tes
<p>Lorem Ipsum è un tes
```

che risulta essere:

Lorem Ipsum è un testo segnaposto utilizzato nel settore della tipografia e della stampa. Lorem Ipsum è considerato il testo segnaposto standard sin dal sedicesimo secolo, quando un anonimo tipografo prese una cassetta

di caratteri e li assemblò per preparare un testo campione.
Lorem Ipsum è un testo segnaposto utilizzato nel settore della tipografia e della stampa. Lorem Ipsum è considerato il testo segnaposto standard sin dal

sedicesimo secolo, quando un anonimo tipografo prese una cassetta di caratteri e li assemblò per preparare un testo campione.

11.2 Stile prima lettera

E' possibile
assegnare uno stile
alla prima lettera
di testo del tag di
ogni elemento
indicato

```
p::first-letter{color:
```

```
<p>Lorem Ipsum è un tes
<p>Lorem Ipsum è un tes
```

che risulta essere:
Lorem Ipsum è un
testo..

Lorem Ipsum è un testo..

11.3 Contenuto prima o dopo un elemento

E' possibile aggiungere contenuto prima o

dopo l'elemento indicato.

Per far ciò si usa ::after e ::before, usabili anche singolarmente.

```
p::after{content:"Prim
p::before{content:"Dop
```

```
<p>Testo..</p>
```

che risulta essere:

Testo

11.4

Selezione

Nel momento in cui l'utente seleziona il testo ` possibile assegnare uno stile.

```
p::selection{color:red
```

```
<p>Testo..</p>
```

che risulta essere:

Testo

12.

COMBINATORI

Ci sono diversi tipi di combinatori, nei paragrafi successivi li esamineremo tutti.

12.1 Selettore Generale

Il più semplice selettore che possiamo immaginare di utilizzare è quello generale, che ha come scope tutti gli elementi HTML della pagina
Questo tipo di selettore è indicato dal carattere * .
Quindi se volessi tutti gli elementi

della pagina di rosso mi basterebbe scrivere nel tag <style> di <head>

```
* {"color:red"}
```

12.2 Selettore Semplice e

Raggruppamento

Abbiamo esaminato diversi esempi di selettore semplice e anche uno di raggruppamento.

Il raggruppamento avviene separando i selettori tramite la virgola. In questo modo gli elementi avranno

lo stesso stile

E' possibile definire

poi anche dello

stile aggiuntivo per

un singolo

selettore

```
h1,h2 { color : blue; t
h1 { font-style:italic
```

12.3

Selettori

Contestuali

A partire da questo punto gli strumenti che esaminiamo permettono di avere un controllo fine sugli elementi del documento
Analizziamo i selettori contestuali (o discendenti).
Permettono di selezionare (e quindi formattare)

tutti gli elementi
che sono
discendenti di un
determinato
elemento
Sono nella forma:

```
selettore1  selettore2  {prop
```

Quindi ad esempio
per avere tutte le
ancore (il tag `a`)
che sono racchiusi
in un tag `p` di
colore blu

scriviamo una regola del tipo

```
p a {color:blue}
```

ottenendo cosi:

Esempio Link di colore blu

12.4 Selettore Figlio

Analizziamo i

selettori figli.
Permettono di
selezionare (e
quindi formattare)
tutti gli elementi
che sono figli di un
determinato
elemento. A
differenza del tipo
precedente dove il
tag può essere
innestato a
qualsiasi livello, in
questo caso deve

essere
immediamente
successivo al
genitore
Sono nella forma:

selettore1 > selettore2 {pr

Quindi ad esempio
per avere tutte le
ancore (il tag a)
che sono racchiusi
immediamente in
un tag p di colore
blu scriviamo una

regola del tipo

```
p > a {color:blue}
```

Una regola del genere influenzerà questo snippet (pezzo di codice):

```
<p> Esempio di  <a> anc
```

ottenendo cosi:

Esempio Link di colore blu

La regola non influenzerà però il

seguente codice

poichè <a> NON

è direttamente

figlio di <p>

```
<p><b> Esempio di </b><
```

12.5 Selettore Fratelli Adiacenti

Analizziamo i

selettori fratelli. Permettono di selezionare (e quindi formattare) tutti gli elementi che sono figli di uno stesso elemento e sono fratelli tra loro. Si applica quando gli elementi hanno lo stesso genitore e il primo tag precede immediatamente il

secondo tag

Sono nella forma:

```
selettore1 + selettore2 {pr
```

Quindi ad esempio per avere tutte le ancore (il tag `a`) che sono racchiusi immediamente in un tag `p` di colore blu scriviamo una regola del tipo

```
b + a {color:blue}
```

```
a + em {color:red}
```

Una regola del genere influenzerà questo snippet (pezzo di codice):

```
<p><b> Esempio di </b><
```

In questo caso quindi i tag <b>, <a> sono tra loro fratelli poichè tutti figli di <p> . In

particolare

abbiamo che

<b>,<a> sono

fratelli adiacenti

cosi come <a>

Otteniamo cosi:

Esempio di testo

enfatizzato a caso

13.

SELETTORE ATTRIBUTI

Con i selettori di attributi si seleziona un elemento in base ai suoi attributi ed ai loro valori. Ci sono quattro modi per specificare i selettori di

attributi. Verranno selezionati gli elementi che corrispondono alle specifiche

13.1. Selettore[attribute

Selezionano gli elementi che hanno quell'attributo qualsiasi sia il

valore di tale attributo.Sono nella forma:

[attribute]

Esempio

```
h1 [title] {color:yell
```

seleziona gli <h1> con attributo title

```
<h1 title="Titolo del tag">es
```

13.2.

Selettore[attribute

Selezionano gli elementi che hanno quell'attributo con il valore indicato.
Sono nella forma:

[attribute=value]

Esempio

```
h1 [title="Titolo"] {c
```

seleziona gli

<h1> con

attributo

title = "Titolo"

Non viene

selezionato ad

esempio

```
<h1  title="Titolo del tag">es
<h1  title="Titolo ">esempio
<h1  title="TITOLO">esempi
```

ma viene

selezionato solo

l'esatta

corrispondenza :

```
<h1  title="Titolo">esempio
```

13.3.

Selettore[attribute

Selezionano gli elementi che tra i valori dell'attributo (separata da spazi) abbia la parola da noi indicata. Sono nella forma:

[attribute~=value]

Esempio

```
h1 [title~="Titolo"] {
```

seleziona gli <h1> con attributo title nel cui value ci sia la parola "Titolo" > esempio h1 </h1>

Esempio:

```
<h1 title="Titolo">esempio
```

```
<h1  title="Titolo del tag">es
<h1  title="Tag Titolo">esem
```

13.4.

Selettore[attribute

Selezionano gli elementi che tra i valori dell'attributo (separata dai trattini) abbia la parola da noi indicata. Sono nella forma:

[attribute|=value]

Esempio

```
h1 [title|="Titolo"] {
```

seleziona gli
<h1> con
attributo title nel
cui value ci sia la
parola "Titolo"
con trattini

Esempio:

```
<h1 title="Titolo"
<h1 title="Titolo-del-tag">es
```

```
<h1  title="Tag-Titolo">esem
```

13.5.

Selettore[attribute

Selezionano gli elementi che abbia come valore iniziale dell'attributo la parola da noi indicata. Sono nella forma:

[attribute^=value]

Esempio

```
h1 [title^="Titolo"] {
```

seleziona gli
<h1> con
attributo title il
cui value inizia per
"Titolo"

Esempio:

```
<h1 title="Titolo"
<h1 title="TitoloTag">esemp
<h1 title="Titolo-Tag">esem
```

```
<h1  title="Titolo Tag">esem
```

13.6.

Selettore[attribute

Selezionano gli elementi che abbia come valore finale dell'attributo la parola da noi indicata. Sono nella forma:

[attribute$=value]

Esempio

```
h1 [title$="Titolo"] {
```

seleziona gli <h1> con attributo title il cui value finisce per "Titolo"

Esempio:

```
<h1 title="Titolo"
<h1 title="Tag Titolo">esem
<h1 title="TagTitolo">esemp
```

13.7.

Selettore[attribute

Selezionano gli elementi che abbia come valore (in qualsiasi posizione) dell'attributo la parola da noi indicata. Sono nella forma:

[attribute*=value]

Esempio

```
h1 [title*="Titolo"] {
```

seleziona gli
<h1> con
attributo title il
cui value finisce
per "Titolo"
Esempio:

```
<h1 title="Titolo">esempio
<h1 title="Tag Titolo TAG">e
<h1 title="TagTitolo">esemp
<h1 title="Titolo-TAG">esem
```