HTML, CSS e JavaScript

Prof.ssa Lucia Tattoli
Prof.ssa Maria Teresa Tattoli

SOMMARIO

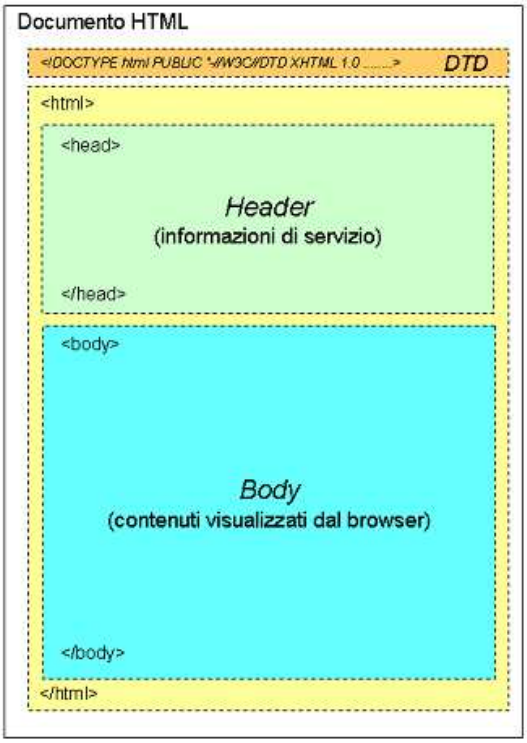
Pagine web e HTML

HTML è acronimo di **Hyper Text Mark-Up Language** e indica il linguaggio usato per descrivere i documenti ipertestuali disponibili nel Web.
Tutti i siti web presenti su Internet sono scritti in codice HTML.

HTML non è un linguaggio di programmazione, ma un sistema di markup, poiché si basa su codici di marcatura, detti **tag** che, inseriti in un testo, ne determinano le caratteristiche di formattazione.
Il browser legge ed elabora il testo in base ai tag, generando la pagina web come noi la vediamo.
Un tag è un carattere o una breve parola racchiusa tra due parentesi angolari, cioè i segni < e >.
Sono tag, ad esempio, <font>, , <br>, ecc..
Generalmente le informazioni su cui agisce il tag devono essere racchiuse fra un tag di apertura ed uno di chiusura, quest'ultimo indicato apponendo il carattere slash (/) dopo la parentesi angolare aperta.
Ad esempio, il testo tra i tag ... appare in grassetto.
I tag possono essere scritti sia in maiuscolo sia in minuscolo, anche se è consigliabile mantenere la coerenza (usare solo i caratteri in maiuscolo o solo in minuscolo).

L'HTML è un linguaggio di pubblico dominio, la cui sintassi è stabilita dal **World Wide Web Consortium** (**W3C**), che ha rilasciato diverse versioni di questo linguaggio: HTML 2.0, HTML 3.2, HTML 4.01 e l'ultimissima HTML 5.
La versione **4.01** (24 dicembre 1999), per anni lo standard più diffuso, sta lentamente lasciando posto alla versione **5**.
Le notevoli novità introdotte da HTML 5 mirano essenzialmente al disaccoppiamento tra struttura e contenuti della pagina. Il World Wide Web Consortium ha annunciato che per il 2016 è prevista l'uscita di HTML 5.1.
Nella presente trattazione si farà riferimento alla versione 4.01, precisando eventuali novità introdotte da HTML 5.

Struttura di una pagina



Una pagina HTML è costituita da solo testo, pertanto può essere scritta usando un semplice elaboratore di testi come il **Blocco note**.
Molti preferiscono però usare speciali editor, in grado di evidenziare il risultato visivo piuttosto che i codici; tali editor sono indicati con il nome di **WYSIWYG** (What You See Is What You Get – Quello che vedi è quello che ottieni).
I più diffusi editor WYSIWIG sono **Visual WEB Developer di Microsoft** e **Dreamweaver di Macromedia.**

A sinistra è riportata la struttura di una semplice pagina.

Un documento HTML comincia con l'indicazione della definizione del tipo di documento (**Document Type Definition** o **DTD**), la quale segnala al browser le specifiche del documento, indicando a quale versione di HTML si fa riferimento. Non è indispensabile, ma il W3C consiglia fortemente di inserirla.

A questo scopo si usa il tag
<!DOCTYPE>
La riga di codice assume un aspetto di questo genere:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" http://www.w3.org/TR/html4/loose.dtd>

Questa riga di codice (che, per comodità, ometteremo nel seguito della trattazione) fornisce alcune informazioni sul documento:

Elemento	Descrizione
HTML	il tipo di linguaggio utilizzato è l'HTML
PUBLIC	il documento è pubblico
W3C	il documento fa riferimento alle specifiche rilasciate dal W3C
-	(segno "meno") le specifiche non sono registrate all'ISO (organizzazione di standardizzazione internazionale). Se lo fossero state, ci sarebbe stato un "+"
DTD HTML 4.01 Transitional	il documento fa riferimento a una DTD ("Document Type Definition" cioè "Definizione del tipo di documento"); la versione di HTML supportata è la 4.01 "transitional"
EN	la lingua con cui è scritta la DTD è l'inglese (IT se è scritta in Italiano)
http://www.w3.org/TR/html4/loose.dtd	indirizzo di riferimento a cui è possibile trovare la DTD. Non è un parametro necessrio, anzi per l'HTML non lo si fa quasi mai, perché gli URL a cui trovare la documentazione sono universalmente noti.

La riga DTD può comunque essere anche scritta semplicemente così:

<!DOCTYPE HTML>

Dopo la riga DTD, inizia la pagina html vera e propria, racchiusa tra il tag **<html>** e la sua chiusura **</html>**.
All'interno ci sono due sezioni: l'intestazione (**header**) e il corpo della pagina (**body**).

L'intestazione, racchiusa fra i tag **<head>** e **</head>**, contiene altri tag che forniscono informazioni supplementari, quali il titolo e indicazioni per rendere la pagina reperibile dai motori di ricerca.
Un tag importante presente nell'intestazione è **<title>**, che serve per specificare il testo che deve comparire nella barra del titolo della finestra del browser; la chiusura **</title>** serve per indicare la fine di questa porzione di testo.

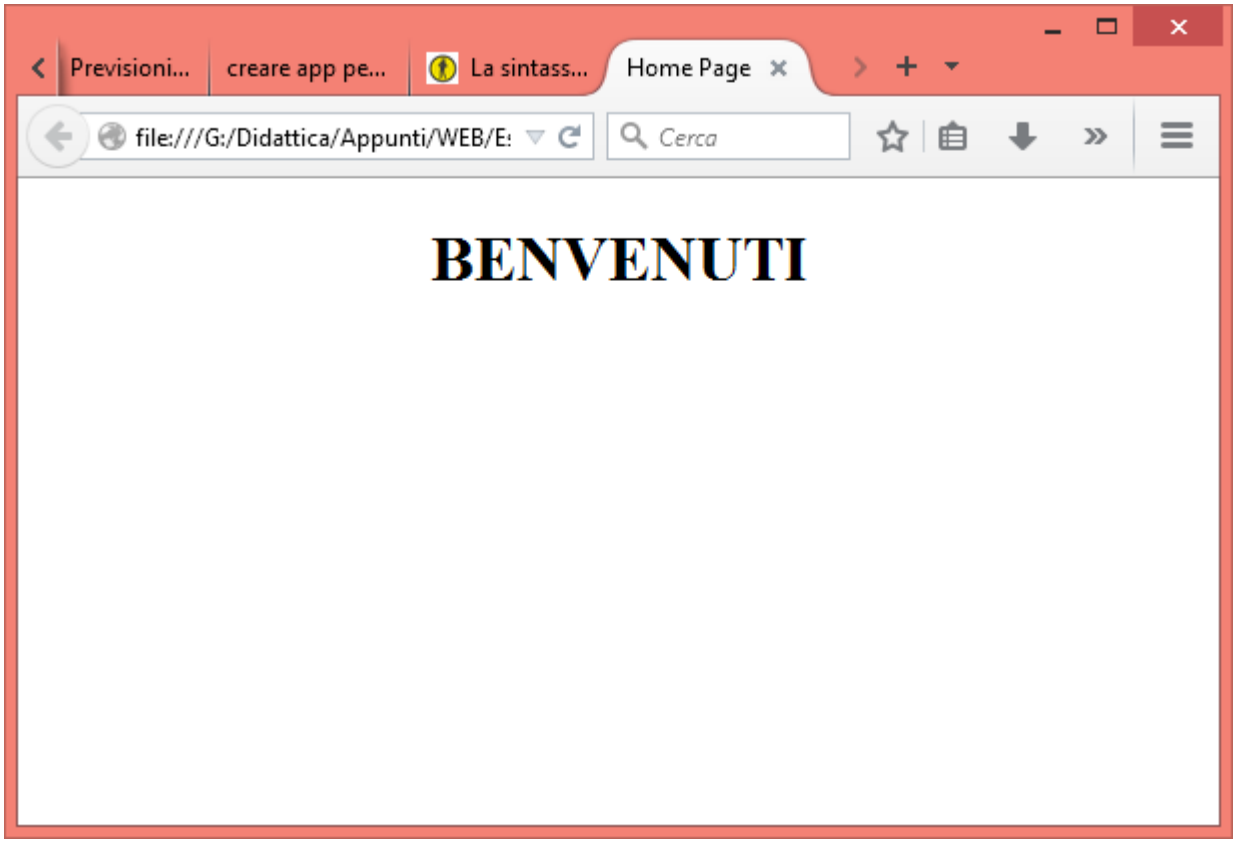
Il corpo della pagina, racchiuso fra i tag **<body>** e **</body>**, contiene tutto ciò che viene poi visualizzato dal browser.

Esempio:

```
<!doctype HTML>
<html>
   <head>
     <title>Home Page</title>
   </head>
   <body>
     <h1 align="center">BENVENUTI</h1>
   </body>
</html>
```

Compare una pagina con un'intestazione al centro della prima riga (BENVENUTI). Il titolo viene inserito all'interno dei tag **<h1>** e **</h1>**, che impostano un'intestazione di dimensione 1, che è quella più grande. L'HTML dispone di sei livelli, da **h1** a **h6**, di intestazione, caratterizzati, oltre che da una differente dimensione del carattere, anche da un diverso utilizzo del grassetto, per cui

il tag <h1> è quello che fornisce l'effetto di maggiore impatto, mentre il tag <h6> ha la rilevanza minore.



I tag possono contenere uno o più **attributi**, che definiscono ulteriori caratteristiche e formattazioni da applicare al testo. Un attributo viene posto dopo il nome del tag, seguito dal segno = e, tra virgolette, dal valore da assegnare all'attributo. Nell'esempio è stato usato l'attributo **align**, cui è stato dato il valore **center**, per visualizzare il titolo al centro della riga.

OSS.: l'indentazione (incolonnamento) delle righe di codice non è obbligatoria, ma può essere utile per migliorare la leggibilità.

Oltre ai tag di base, per poter creare una pagina HTML riconosciuta come valida nella versione 5, è necessario inserire nel codice alcune informazioni aggiuntive:

```
<!doctype HTML>
<html lang="it">
   <head>
      <meta charset="UTF-8" />
      <title>Home Page</title>
   </head>
   <body>
      <h1 align="center">BENVENUTI</h1>
   </body>
</html>
```

- **lang** è l'attributo del tag <html> che definisce la lingua del documento (it = Italiano).

- Il tag **<meta />** fornisce meta-informazioni invisibili gli utenti, ma utili per la classificazione della pagina web e importanti per la indicizzazione corretta all'interno dei motori di ricerca, come set dei caratteri, autore, parole chiave della pagina, ecc. In questo caso, attraverso l'attributo **charset**, definisce l'insieme dei caratteri usati per scrivere nella lingua della pagina. UTF-8 è uno standard che utilizza 8 bit per la memorizzazione di un carattere.

I colori in HTML

I colori in HTML si possono esprimere in due modi:

- in formato **RGB** (**R**ed **G**reen **B**lue);
- indicandone il nome.

Il primo si basa sulla combinazione dei tre colori fondamentali Rosso, Verde e Blu per ottenere un qualsiasi colore nella gamma di quelli disponibili. L'intensità di ogni componente cromatico è rappresentata da un numero compreso tra 0 e 255 (in esadecimale tra 00 e FF). Si ottengono in questo modo codici di sei cifre esadecimali, che vanno sempre preceduti dal simbolo # e compresi tra apici.

La seconda possibilità, non compatibile con tutti i browser, permette di indicare direttamente il nome del colore desiderato (naturalmente in inglese).

Si riportano di seguito i codici esadecimali e i nomi dei 126 colori fondamentali (per poterli visualizzare, la scheda grafica deve supportare almeno 256 colori).

"#A0CE00"	"ALICEBLUE"
"#00FFFF"	"AQUA"
"#F0FFFF"	"AZURE"
"#FFE4C4"	"BISQUE"
"#FFEBCD"	"BLANCHEDALMOND"
"#8A2BE2"	"BLUEVIOLET"
"#DEB887"	"BURLYWOOD"
"#7FFF00"	"CHARTREUSE"
"#FF7F50"	"CORAL"
"#FFF8DC"	"CORNSILK"
"#00FFFF"	"CYAN"
"#483D8B"	"DARKSLATEBLUE"
"#B8860B"	"DARKGOLDENROD"
"#FF1493"	"DEEPPINK"
"##696969"	"DIMGRAY"
"#822222"	"FIREBRICK"
"#228B22"	"FORESTGREEN"
"#DCDCDC"	"GAINSBORO"
"#FFD700"	"GOLD"
"#808080"	"GRAY"
"#ADFF2F"	"GREENYELLOW"
"#FF69B4"	"HOTPINK"
"#4B0082"	"INDIGO"
"#F0E68C"	"KHAKY"
"#FFF0F5"	"LAVENDERBLUSH"
"#ADD8E6"	"LIGHTBLUE"
"#E0FFFF"	"LIGHTCYAN"
"#90EE90"	"LIGHTGREEN"
"#FFB6C1"	"LIGHTPINK"

"#FAEBD7"	"ANTIQUEWHITE"
"#7FFFD4"	"AQUAMARINE"
"#F5F5DC"	"BEIGE"
"#000000"	"BLACK"
"#0000FF"	"BLUE"
"#A52A2A"	"BROWN"
"#5F9EA0"	"CADETBLUE"
"#D2691E"	"CHOCOLATE"
"#6495ED"	"CORNFLOWERBLUE"
"#DC143C"	"CRIMSON"
"#00008B"	"DARKBLUE"
"##008B8B"	"DARKCYAN"
"#A9A9A9"	"DARKGRAY"
"#00BFFF"	"DEEPSKYBLUE"
"#1E90FF"	"DODGERBLUE"
"#FFFAF0"	"FLORALWHITE"
"#FF00FF"	"FUCHSIA"
"#F8F8FF"	"GHOSTWHITE"
"#DAA520"	"GOLDENROD"
"#008800"	"GREEN"
"#F0FFF0"	"HONEYDEW"
"#CD5C5C"	"INDIANRED"
"#FFFFF0"	"IVORY"
"#E6E6FA"	"LAVENDER"
"#FFFACD"	"LEMONCHIFFON"
"#F08080"	"LIGHTCORAL"
"#FAFAD2"	"LIGHTGOLDENRODYELLOW"
"#D3D3D3"	"LIGHTGRAY"
"#FFA07A"	"LIGHTSALMON"

"#20B2AA"	"LIGHTSEAGREEN"
"#778899"	"LIGHTSLATEGRAY"
"#FFFFE0"	"LIGHTYELLOW"
"#32CD32"	"LIMEGREEN"
"#FF00FF"	"MAGENTA"
"#66CDAA"	"MEDIUMAQUAMARINE"
"#BA55D3"	"MEDIUMMORCHID"
"#3CB371"	"MEDIUMSEAGREEN"
"#00FA9A"	"MEDIUMSPRINGGREEN"
"#C71585"	"MEDIUMVIOLETRED"
"#F5FFFA"	"MINTCREAM"
"#FFDEAD"	"NAVAJOWHITE"
"#FDF5E6"	"OLDLACE"
"#6B8E23"	"OLIVEDRAB"
"#FF4500"	"ORANGERED"
"#EEE8AA"	"PALEGOLDENROD"
"#AFEEEE"	"PALETURQUOISE"
"#FFEFD5"	"PAPAYAWHIP"
"#CD853F"	"PERU"
"#DDA0DD"	"PLUM"
"#800080"	"PURPLE"
"#BC8F8F"	"ROSYBROWN"
"#8B4513"	"SADDLEBROWN"
"#F4A460"	"SANDYBROWN"
"#FFF5EE"	"SEASHELL"
"#C0C0C0"	"SILVER"
"#6A5ACD"	"SLATEBLUE"
"#FFFAFA"	"SNOW"
"#468284"	"STEELBLUE"
"#008080"	"TEAL"
"#FF6347"	"TOMATO"
"#EE82EE"	"VIOLET"
"#FFFFFF"	"WHITE"
"#FFFF00"	"YELLOW"

"#87CEFA"	"LIGHTSKYBLUE"
"#B0C4DE"	"LIGHTSTEELBLUE"
"#00FF00"	"LIME"
"#FAF0E6"	"LINEN"
"#800000"	"MAROON"
"#0000CD"	"MEDIUMBLUE"
"#9370DB"	"MEDIUMPURPLE"
"#7B68EE"	"MEDIUMSLATEBLUE"
"#48D1CC"	"MEDIUMTORQUOISE"
"#191970"	"MIDNIGHTBLUE"
"#FFE4E1"	"MISTYROSE"
"#000080"	"NAVY"
"#808000"	"OLIVE"
"#FFA500"	"ORANGE"
"#DA70D6"	"ORCHID"
"#98FB98"	"PALEGREEN"
"#DB7093"	"PALEVIOLETRED"
"#FFDAB9"	"PEACHPUFF"
"#FFC0CB"	"PINK"
"#B0E0E6"	"POWDERBLUE"
"#FF0000"	"RED"
"#4169E1"	"ROYALBLUE"
"#FA8072"	"SALMON"
"#2E8B57"	"SEAGREEN"
"#A0522D"	"SIENNA"
"#87CEEB"	"SKYBLUE"
"#708090"	"SLATEGRAY"
"#00FF7F"	"SPRINGGREEN"
"#D2B48C"	"TAN"
"#D8BFD8"	"THISTLE"
"#40E0D0"	"TURQUOISE"
"#F5DEB3"	"WHEAT"
"#F5F5F5"	"WHITESMOKE"
"#9ACD32"	"YELLOWGREEN"

Oltre a questi colori fondamentali è possibile ottenere tantissime altre combinazioni.
Ad esempio, per le diverse tonalità del blu si possono usare:

"#0000CC"
"#000088"
"#000055"
"#000022"

ma tutte le combinazioni possibili sono supportate solo da schede grafiche con 16M di colori.

Attributi del tag <body>

Oltre all'attributo **lang**, il tag **<body>** dispone di diversi attributi. I principali sono:

- **bgcolor**, che consente di impostare un colore di sfondo per la pagina.

  Esempio:
  <body bgcolor = "#FF0000">
  <body bgcolor = "red">
  Impostano entrambi lo sfondo rosso alla pagina.

- **background**, che consente di inserire una texture (motivo di riempimento o immagine di qualunque tipo) come sfondo della pagina; la texture può essere un file in formato GIF, JPEG oppure PNG.

  Esempio: <body background="sfondo.gif">

  Poiché non è definito alcun path, lo sfondo viene cercato nella stessa directory del documento HTML.
  L'immagine adoperata a riempimento può essere tenuta ferma durante lo spostamento verticale (scrolling) sulla pagina, dando l'impressione di scivolare con le immagini e i testi sullo sfondo, operazione per altro possibile soltanto con i browser Internet Explorer.

  Esempio: <body background="nome_immagine.gif" bgproperties="fixed">

- **text**, che consente di impostare il colore del testo della pagina.

- **link**, che consente di impostare il colore dei link (collegamenti) presenti nella pagina.

- **vlink**, che consente di impostare il colore dei link visitati presenti nella pagina.

- **alink**: che consente di impostare il colore dei link attivi presenti nella pagina.

- **topmargin, bottommargin, leftmargin, rightmargin**: sono i quattro attributi per definire la distanza in pixel rispettivamente da margine superiore, margine inferiore, margine sinistro e margine destro. Un margine uguale a zero farà si che il testo inizi praticamente sul bordo della finestra del browser.
  Sono attributi proprietari di alcuni browser non riconosciuti dal W3C.

Riassumendo, quella che segue è la definizione del corpo di una pagina con un'immagine di sfondo che resta fissa durante le operazioni di spostamento (scrolling), testo di colore nero, link di colore rosso, link visitati di colore verde e distanza dai quattro margini di 50 pixel.

<body background=" immaginesfondo.gif" bgproperties="fixed" text="black" link="red" vlink="green" topmargin="50" bottommargin="50" leftmargin="50" rightmargin="50">

Le entità carattere

A volte in un documento HTML è necessario usare caratteri speciali, come lettere accentate, simboli monetari (quelli sulla tastiera non sempre sono comprensibili a tutti i browser), caratteri non disponibili sulla tastiera. Tra i caratteri speciali sono presenti anche quelli riservati all'HTML, quali i simboli "<", ">", "&" e le virgolette. Anche lo spazio è un carattere speciale: **il browser ignora più di uno spazio tra due parole**.
In HTML esistono le cosiddette **entità**, ossia costrutti sintattici che permettono ai browser di far visualizzare correttamente questi caratteri. Cosi come i tag, anche le entità sono delimitate da due caratteri: "**&**" e "**;**". Tra delimitatori vengono inseriti i codici numerici (codici ISO) o i nomi simbolici (appositamente definiti per l'HTML) che corrispondono ai caratteri speciali.

Entità per caratteri ISO 8859-1

	&iexcl;	&cent;	&pound;	&curren;	&yen;	&brvbar;	&sect;	&uml;	&copy;	&ordf;	&laquo;
NB SP	¡	¢	£	¤	¥	¦	§	¨	©	a	«
	&#161;	&#162;	&#163;	&#164;	&#165;	&#166;	&#167;	&#168;	&#169;	&#170;	&#171;
&not;	&shy;	&reg;	&macr;	&deg;	&plusmn;	&sup2;	&sup3;	&acute;	&micro;	&para;	&middot;
¬	-	®	¯	°	±	2	3	´	µ	¶	·
&#172;	&#173;	&#174;	&#175;	&#176;	&#177;	&#178;	&#179;	&#180;	&#181;	&#182;	&#183;
&cedil;	&sup1;	&ordm;	&raquo;	&frac14;	&frac12;	&frac34;	&iquest;	&Agrave;	&Aacute;	&Acirc;	&Atilde;
¸	1	o	»	¼	½	¾	¿	À	Á	Â	Ã
&#184;	&#185;	&#186;	&#187;	&#188;	&#189;	&#190;	&#191;	&#192;	&#193;	&#194;	&#195;
&Auml;	&Aring;	&AElig;	&Ccedil;	&Egrave;	&Eacute;	&Ecirc;	&Euml;	&Igrave;	&Iacute;	&Icirc;	&Iuml;
Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
&#196;	&#197;	&#198;	&#199;	&#200;	&#201;	&#202;	&#203;	&#204;	&#205;	&#206;	&#207;
&ETH;	&Ntilde;	&Ograve;	&Oacute;	&Ocirc;	&Otilde;	&Ouml;	&times;	&Oslash;	&Ugrave;	&Uacute;	&Ucirc;
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û
&#208;	&#209;	&#210;	&#211;	&#212;	&#213;	&#214;	&#215;	&#216;	&#217;	&#218;	&#219;
&Uuml;	&Yacute;	&THORN;	&szlig;	&agrave;	&aacute;	&acirc;	&atilde;	&auml;	&aring;	&aelig;	&ccedil;
Ü	Ý	Þ	ß	à	á	â	ã	ä	å	æ	ç
&#220;	&#221;	&#222;	&#223;	&#224;	&#225;	&#226;	&#227;	&#228;	&#229;	&#230;	&#231;
&egrave;	&eacute;	&ecirc;	&euml;	&igrave;	&iacute;	&icirc;	&iuml;	&eth;	&ntilde;	&ograve;	&oacute;
è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó
&#232;	&#233;	&#234;	&#235;	&#236;	&#237;	&#238;	&#239;	&#240;	&#241;	&#242;	&#243;
&ocirc;	&otilde;	&ouml;	&divide;	&oslash;	&ugrave;	&uacute;	&ucirc;	&uuml;	&yacute;	&thorn;	&yuml;
ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ
&#244;	&#245;	&#246;	&#247;	&#248;	&#249;	&#250;	&#251;	&#252;	&#253;	&#254;	&#255;

Entità per simboli matematici

&bull;	&hellip;	&prime;	&Prime;	&oline;	&frasl;	&weierp;	&image;	&real;	&trade;	&alefsym;	&larr;
•	…	′	″	‾	⁄	℘	ℑ	ℜ	™	ℵ	←
&#8226;	&#8230;	&#8242;	&#8243;	&#8254;	&#8260;	&#8472;	&#8465;	&#8476;	&#8482;	&#8501;	&#8592;
&uarr;	&rarr;	&darr;	&harr;	&crarr;	&lArr;	&uArr;	&rArr;	&dArr;	&hArr;	&forall;	&part;
↑	→	↓	↔	↵	⇐	⇑	⇒	⇓	⇔	∀	∂
&#8593;	&#8594;	&#8595;	&#8596;	&#8629;	&#8656;	&#8657;	&#8658;	&#8659;	&#8660;	&#8704;	&#8706;
&exist;	&empty;	&nabla;	&isin;	&notin;	&ni;	&prod;	&sum;	&minus;	&lowast;	&radic;	&prop;
∃	∅	∇	∈	∉	∋	∏	∑	−	∗	√	∝
&#8707;	&#8709;	&#8711;	&#8712;	&#8713;	&#8715;	&#8719;	&#8721;	&#8722;	&#8727;	&#8730;	&#8733;
&infin;	&ang;	&and;	&or;	&cap;	&cup;	&int;	&there4;	&sim;	&cong;	&asymp;	&ne;
∞	∠	∧	∨	∩	∪	∫	∴	~	≅	≈	≠
&#8734;	&#8736;	&#8743;	&#8744;	&#8745;	&#8746;	&#8747;	&#8756;	&#8764;	&#8773;	&#8776;	&#8800;
&equiv;	&le;	&ge;	&sub;	&sup;	&nsub;	&sube;	&supe;	&oplus;	&otimes;	&perp;	&sdot;
≡	≤	≥	⊂	⊃	⊄	⊆	⊇	⊕	⊗	⊥	·
&#8801;	&#8804;	&#8805;	&#8834;	&#8835;	&#8836;	&#8838;	&#8839;	&#8853;	&#8855;	&#8869;	&#8901;
&lceil;	&rceil;	&lfloor;	&rfloor;	&lang;	&rang;	&loz;	&spades;	&clubs;	&hearts;	&diams;	
⌈	⌉	⌊	⌋	〈	〉	◊	♠	♣	♥	♦	
&#8968;	&#8969;	&#8970;	&#8971;	&#9001;	&#9002;	&#9674;	&#9824;	&#9827;	&#9829;	&#9830;	

Entità per codici di marcatura e internazionalizzazione

&quot;	&amp;	&lt;	&gt;	&OElig;	&oelig;	&Scaron;	&scaron;	&Yuml;	&circ;	&tilde;	
"	&	<	>	Œ	œ	Š	š	Ÿ	ˆ	˜	EN SP
&#34;	&#38;	&#60;	&#62;	&#338;	&#339;	&#352;	&#353;	&#376;	&#710;	&#732;	
		&zwnj;	&zwj;	&lrm;	&rlm;	&ndash;	&mdash;	&lsquo;	&rsquo;	&sbquo;	&ldquo;
EM SP	THIN SP	ZW NJ	ZW J	LRM	RLM	–	—	‘	’	‚	“
		&#8204;	&#8205;	&#8206;	&#8207;	&#8211;	&#8212;	&#8216;	&#8217;	&#8218;	&#8220;
&rdquo;	&bdquo;	&dagger;	&Dagger;	&permil;	&lsaquo;	&rsaquo;	&euro;				
”	„	†	‡	‰	‹	›	€				
&#8221;	&#8222;	&#8224;	&#8225;	&#8240;	&#8249;	&#8250;	&#8364;				

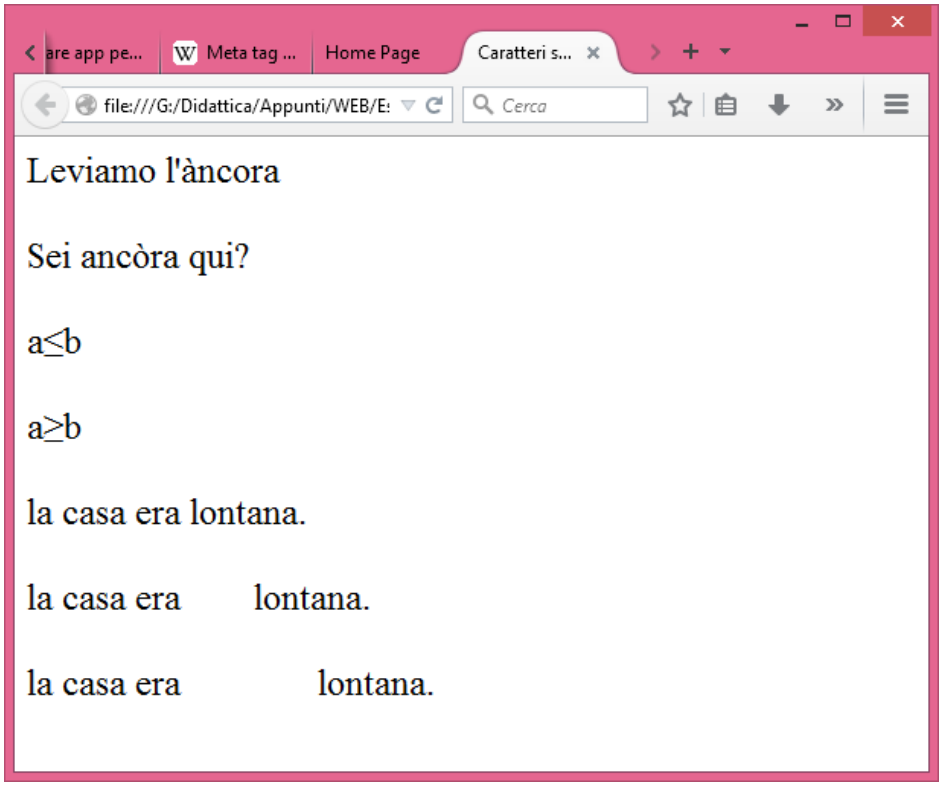
Entità per caratteri greci

&fnof;	&Alfa;	&Beta;	&Gamma;	&Delta;	&Epsilon;	&Zeta;	&Eta;	&Theta;	&Iota;	&Kappa;	&Lambda;
ƒ	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ
&#402;	&#913;	&#914;	&#915;	&#916;	&#917;	&#918;	&#919;	&#920;	&#921;	&#922;	&#923;
&Mu;	&Nu;	&Xi;	&Omicron;	&Pi;	&Rho;	&Sigma;	&Tau;	&Upsilon;	&Phi;	&Chi;	&Psi;
Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ
&#924;	&#925;	&#926;	&#927;	&#928;	&#929;	&#931;	&#932;	&#933;	&#934;	&#935;	&#936;
&Omega;	&alfa;	&beta;	&gamma;	&delta;	&epsilon;	&zeta;	&eta;	&theta;	&iota;	&kappa;	&lambda;
Ω	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ
&#937;	&#945;	&#946;	&#947;	&#948;	&#949;	&#950;	&#951;	&#952;	&#953;	&#954;	&#955;
&mu;	&nu;	&xi;	&omicron;	&pi;	&rho;	&sigmaf;	&sigma;	&tau;	&upsilon;	&phi;	&chi;
μ	ν	ξ	ο	π	ρ	ς	σ	τ	υ	φ	χ
&#956;	&#957;	&#958;	&#959;	&#960;	&#961;	&#962;	&#963;	&#964;	&#965;	&#966;	&#967;
&psi;	&omega;	&thetasym;	&upsih;	&piv;							
ψ	ω	ϑ	ϒ	ϖ							
&#968;	&#969;	&#977;	&#978;	&#982;							

Esempio:

```
<!doctype HTML>
<html>
   <head>
      <title>
         Caratteri speciali
      </title>
   </head>
   <body>
      <font size="5">
      Leviamo l'&agrave;ncora
      <br><br>
      Sei anc&ograve;ra qui?
      <br><br>
      a&le;b
      <br><br>
      a&ge;b
      <br><br>
      la casa era                lontana.
      <br><br>
      la casa era        lontana.
```

```
        <br><br>
        la casa era               lontana.
        <br>
        </font>
    </body>
</html>
```



Si osservi che, per inserire più di uno spazio tra due parole, è necessario ripetere più volte l'entità che rappresenta lo spazio, ** ** .
Si osservi anche che nell'esercizio alle pagine 17 e 18, il carattere ** ** è stato usato per "riempire" le celle della riga vuota. Se non fossero stati indicati, infatti, la riga non sarebbe apparsa, in quanto i margini superiore e inferiore sarebbero stati "schiacciati" l'uno sull'altro.

Il percorso dei file

Il percorso di un file a cui far riferimento in una pagina web può essere specificato in due modi:

- **percorso assoluto**;
- **percorso relativo**.

Il percorso assoluto fa riferimento alla posizione che occupa un file all'interno del disco che contiene il sito.
Per esempio, il file *sfondo.gif*, che si trova nella cartella di nome *Immagini* contenuta a sua volta nella cartella di nome *Sito* contenuta nella root del sistema ha come percorso assoluto:

c:/Sito/Immagini/sfondo.gif

Utilizzare questo tipo di percorso non è consigliabile per i file di un sito, perché li lega alla posizione che essi occupano all'interno del disco. Spostando la cartella *Sito* su un'altra macchina, in una posizione diversa dalla root (per esempio sul server che fornisce il servizio di hosting) i file non saranno più raggiungibili.

Conviene usare il percorso relativo, che, come dice la parola, rappresenta il percorso da seguire partendo dalla posizione in cui ci si trova. La posizione di partenza è indicata dal simbolo "./". Stabilita la posizione di partenza, per indicare di salire di un livello si usa il simbolo "../", per scendere di un livello è sufficiente specificare il nome della cartella seguito dal simbolo "/".

Esempi:

./../index.asp	*partendo dalla posizione corrente, si sale di un livello e si trova il file "index.asp";*
./immagini/sfondo.gif	*partendo dalla posizione corrente, si scende nella cartella immagini, dove si trova il file "sfondo.gif";*
./../../immagini/sfondo.gif	*partendo dalla posizione corrente, si sale di due livelli per poi scendere di un livello attraverso la cartella immagini, dove si trova il file "sfondo.gif".*

Come già detto, se non si specifica un path ma solo il nome del file, quest'ultimo viene cercato nella stessa directory del documento HTML.

Il tag font

Il tag consente di cambiare il font dei caratteri racchiusi fra **** e ****.
Tramite l'attributo **face**, si specifica il carattere con il quale il testo deve essere visualizzato.
Si possono elencare diversi nomi di caratteri separati da virgole; se il primo non è disponibile, il browser proverà uno alla volta i successivi, fino a trovare quello che può utilizzare. Se nessuno dei caratteri indicati è disponibile, è usato un carattere predefinito.

Esempio: <font face="arial, times new roman, verdana">

La dimensione del font è impostata tramite l'attributo **size**; la misura è rappresentata da un numero compreso fra 1 e 7.

Esempio: <font size="5">

È possibile specificare una dimensione relativa. Per default, la misura di un carattere è 3; se, come valore dell'attributo size, si usa uno dei valori negativi -1, -2 e -3, il carattere viene ridotto di 1,2 o 3 rispetto alla dimensione di default, mentre, se si usa uno dei numeri positivi +1, +2, +3 e +4, il carattere viene ingrandito di 1, 2, 3 o 4 rispetto al valore di default.

Esempio: <font size="+3">

Il colore del testo può essere impostato tramite l'attributo **color.**
Esempio: <font color="Navy" size="5" face="arial">

Il tag br

Il tag
 (break) inserisce un **ritorno a capo** (**carriage return**). In assenza di questo tag, il testo viene scritto di seguito, fino a raggiungere il lato destro della finestra del browser, anche se nel file html si è usato il tasto Return per andare a capo.
Questo tag non ammette chiusura e, come tutti i tag di questo tipo, si può scrivere anche nel seguente modo:

Esempio: *Questo testo **
** va a capo* → *Questo testo*
va a capo

Il tag p

Il tag ** ** consente di indicare l'inizio di un paragrafo. Ogni paragrafo é individuato graficamente da una riga vuota che lo precede e da una che lo segue. Il tag ** ** individua la chiusura del paragrafo; in realtà esso non è necessario, perché la fine di un paragrafo è individuata dall'inizio del successivo. In ogni caso è buona norma includere tutti i tag di chiusura.
È possibile impostare l'allineamento del testo di un paragrafo rispetto ai margini tramite l'attributo **align**. Le opzioni consentite sono:

- **center** - il testo viene centrato rispetto ai margini della finestra;
- **right** - il testo viene allineato al margine destro della finestra del browser;
- **left** - il testo viene allineato al margine sinistro della finestra del browser;
- **justify** - il testo viene giustificato.

Esempio:

```
<p align="center">questo è un esempio di paragrafo</p>
<p align="center">questo è un altro paragrafo</p>
```

Il tag div

Il tag è un contenitore generico usato per racchiudere una parte di testo cui attribuire caratteristiche comuni.

Ha funzioni analoghe a quelle di , producendo un "a capo" prima e dopo la parte di testo tra e , ma senza lasciare una riga vuota come .
Un attributo importante del tag è **id**, il cui valore è un identificatore univoco che individua la parte della pagina delimitata da e .

Il tag center

Il tag ** ** (chiusura ** **) consente di posizionare il testo al centro della pagina. Non è molto usato; al suo posto si preferisce utilizzare il tag ** ** (chiusura ** **) con attributo **align**.

Esempio

```
<center>testo</center>
<div align="center">testo</div>
```

```
entrambi posizionano il testo a
centro della pagina.
```

I tag per le liste

Si possono usare tre tipi di liste: puntate (o non ordinate), numerate (o ordinate) e di definizione
Il tag **
** indica l'inizio di una **lista puntata**; ogni elemento della lista viene preceduto dal tag **
** e chiuso con **
** (la chiusura non è obbligatoria).

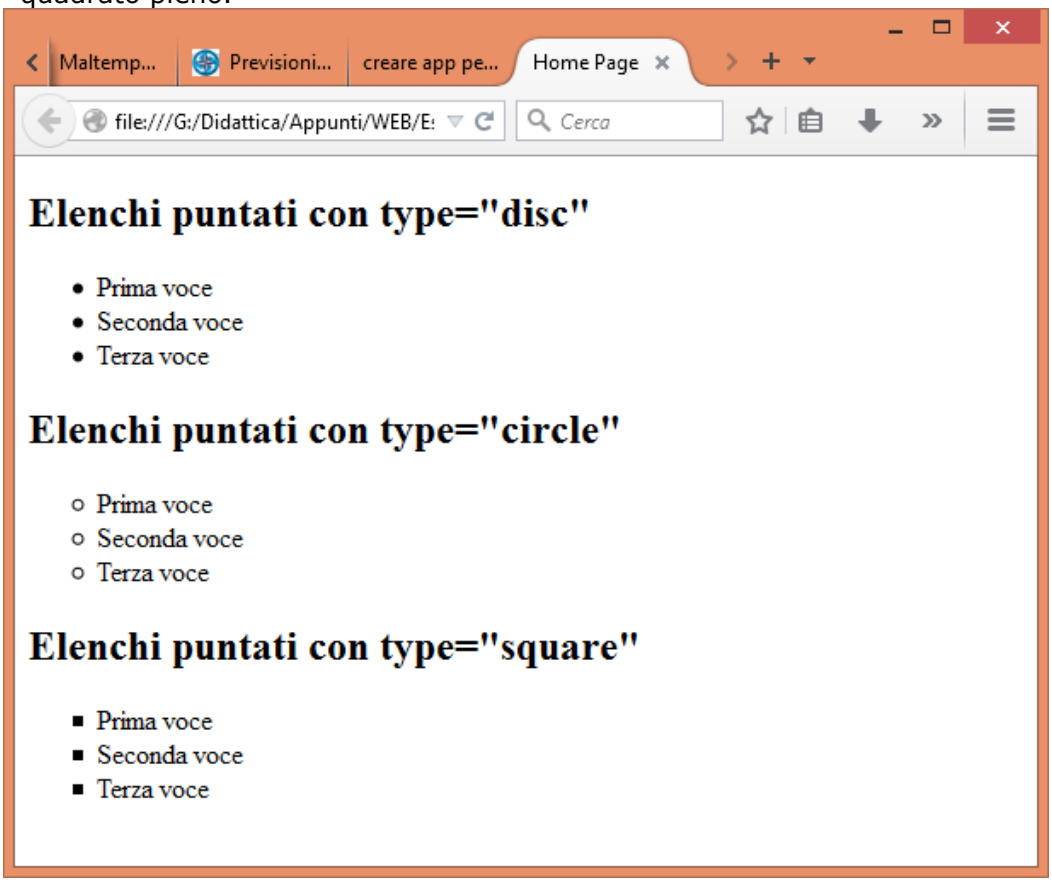
Esempio:

```
<ul>
    <li>Prima voce
    <li>Seconda voce
    <li>Terza voce
</ul>
```

Normalmente ogni voce comparire preceduta da un cerchietto nero. È possibile usare altri simboli attrvarso l'attributo **type** di <ul>; i valori possibili sono:

disc - disco o cerchio (default)

circle - circonferenza
square - quadrato pieno.



Per creare una **lista numerata** si usano i tag **
…
** invece dei tag **
….**

Esempio:

```
<ol type="a">
    <li>prima voce
    <li>seconda voce
    <li>terza voce
</ol>
```

Per cambiare il tipo di numerazione, si può usare l'attributo **type**, che assume i seguenti valori:

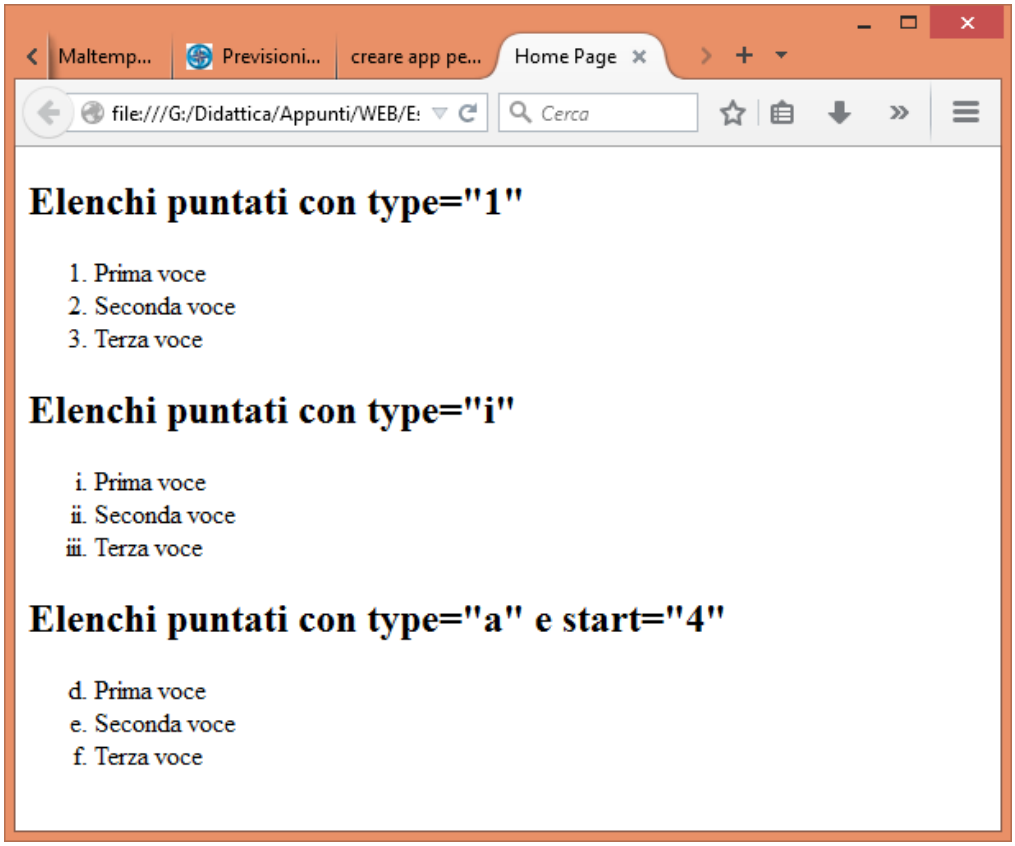
1 - numerazione normale
I - numeri romani
i - numeri romani minuscoli
a - lettere minuscole
A - lettere maiuscole

In alcuni casi potrebbe essere necessario iniziare la numerazione di una lista con un numero diverso dal primo. A tal fine si può usare l'attributo **start**, che permette di impostare il numero di partenza. Il valore di start è il numero d'ordine nella sequenza di lista da cui si intende far partire elencazione (non il simbolo).

Esempio:

<ol type="1" start="4">
produce un elenco individuato dalle lettere minuscole da 4 in poi.

<ol type="a" start="3">
produce un elenco individuato dalle lettere minuscole da c in poi.



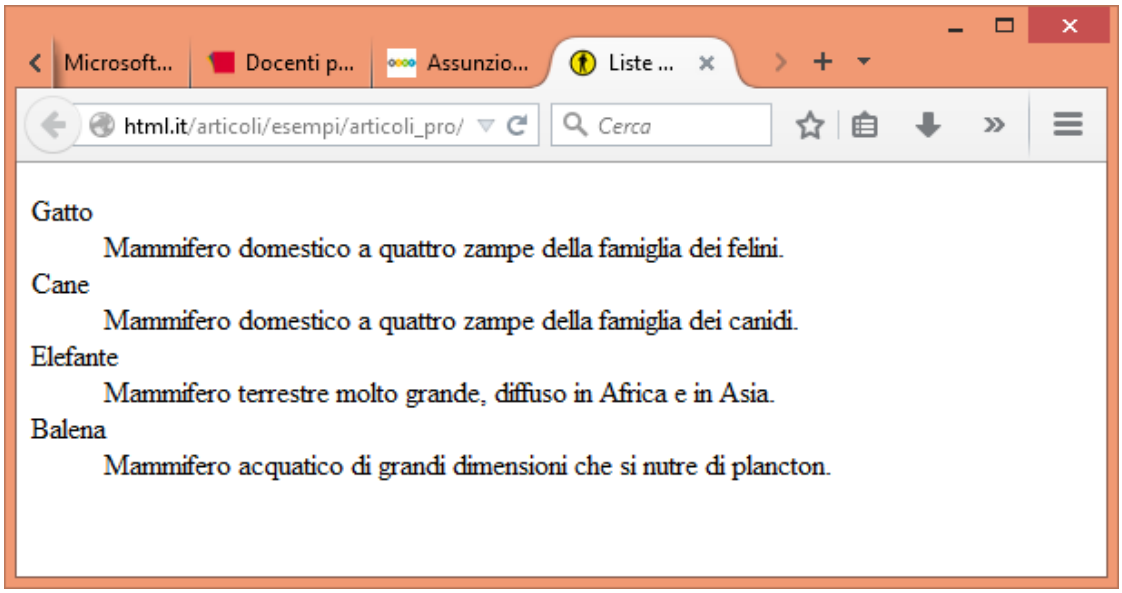
La **lista di definizione** si differenzia dalle precedenti in quanto ogni elemento della lista è formato da due parti:

- un termine
- la definizione di tale termine.

Normalmente la definizione è visualizzata in una riga differente rispetto al termine ed è rientrata rispetto al margine sinistro del testo.

I tag per l'apertura e la chiusura della lista sono: **<dl>** e **</dl>**, i tag per l'inserimento del termine sono **<dt>** e **</dt>** e quelli per la definizione sono **<dd>** e **</dd>**.

Esempio: la seguente lista

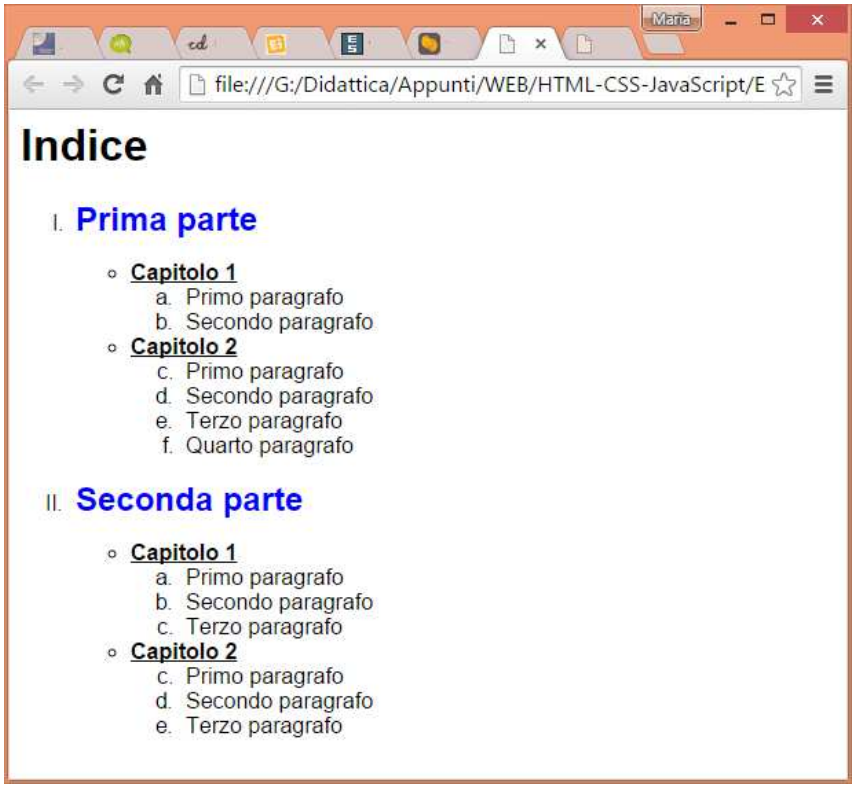


è così prodotta:

```
<dl>
<dt>Gatto</dt>
<dd>Mammifero domestico a quattro zampe della famiglia dei felini.</dd>
<dt>Cane</dt>
<dd>Mammifero domestico a quattro zampe della famiglia dei canidi.</dd>
<dt>Elefante</dt>
<dd>Mammifero terrestre molto grande, diffuso in Africa e in Asia.</dd>
<dt>Balena</dt>
<dd>Mammifero acquatico di grandi dimensioni che si nutre di plancton.</dd>
</dl>
```

Si possono creare anche liste nidificate: basta inserire l'intera struttura di una lista come elemento di un'altra lista. Si consglia in questo caso di chiudere i tag <li> che contengono altre liste.

Esempio: Per la seguente pagina



il codice è

```
<!doctype HTML>
<html>

<head>
<title>liste</title>
</head>
<body>
<font face="Arial">
<h1>Indice</h1>
<ol TYPE="I">
  <li><p><font color="blue" size="5"><b>Prima parte</b></font>
      <ul>
        <li><u><b>Capitolo 1</b></u>
```

```
        <ol TYPE="a">
          <li>Primo paragrafo
          <li>Secondo paragrafo
        </ol>
        <li><u><b>Capitolo 2</b></u>
        <ol TYPE="a" start="3">
          <li>Primo paragrafo
          <li>Secondo paragrafo
          <li>Terzo paragrafo
          <li>Quarto paragrafo
        </ol>
      </ul>
  <li><p><font color="blue" size="5"><b>Seconda parte</b></font>
      <ul>
        <li><u><b>Capitolo 1</b></u>
        <ol TYPE="a">
          <li>Primo paragrafo
          <li>Secondo paragrafo
          <li>Terzo paragrafo
        </ol>
        <li><u><b>Capitolo 2</b></u>
        <ol TYPE="a" start="3">
          <li>Primo paragrafo
          <li>Secondo paragrafo
              <li>Terzo paragrafo

        </ol>
      </ul>
</ol>
</font>
</body>
</html>
```

Tag stilistici e tag di contesto

I tag che attribuiscono stili ai caratteri sono di due tipi:

- tag stilistici (o fisici)

…	corsivo
…	grassetto
<u>…</u>	sottolineato
<big>…</big>	carattere più grande
…	carattere più piccolo
<tt>…</tt>	carattere teletype a spaziatura fissa

- tag di contesto (o logici).

<cite>…</cite>	citazione (corsivo)
<code>…</code>	codice di un programma
…	testo di valore emotivo (corsivo)
<samp>…</samp>	esempio
…	testo che ha peso (grassetto)
<name>…</name>	nome di una persona
<person>…</person>	nome di una persona
<var>…</var>	variabili in un programma
<time>…</time>	per data e ora (l'attributo datetime imposta il formato)
<address>…</address>	indirizzi

I primi conferiscono al testo solo un'importanza tipografica; sono visualizzati sempre nello stesso modo da tutti i browser.
I secondi, oltre a enfatizzare in qualche modo il testo, segnalano al browser anche un'importanza semantica individuata dal tag stesso. Possono essere visualizzati in modo diverso a seconda del browser utilizzato. Ve ne sono molti altri rispetto a quelli indicati, vista l'importanza che il web semantico sta assumendo.

Altri tag che influenzano il formato sono:

[…]	apici	*Esempio:*	*3³=27 *	→	$3^3=27$
_…	pedici	*Esempio:*	*H₂O*	→	H_2O

Commenti

A volte può essere utile commentare il codice HTML. I commenti non sono visualizzati dal browser, ma saranno sicuramente utili quando si dovrà "rimettere mano al codice", per cambiarlo o semplicemente per usarlo in altre occasioni.
I commenti HTML iniziano con **<!--** e finiscono con **-->**. Tutto ciò che è compreso tra questi due marcatori non viene interpretato come codice HTML.

Inserimento di immagini

Il tag per inserire le immagini è **<img>** e non ammette chiusura. L'attributo **src** specifica il percorso del file che contiene l'immagine, che può essere un file di un qualsiasi formato grafico (.jpg, bmp, .png, .gif; l'immagine gif può anche essere animata). La sintassi è dunque:

<img src ="percorso del file che contiene l'immagine">

Se non si specifica alcun percorso, il file viene prelevato dalla directory corrente; nel caso in cui il file si trovi altrove, si deve specficare l'URL o il path completo.

Esempio:

<img src="foto1.gif">	*Il file foto1.gif è nella directory corrente.*
<img src="./immagini/foto1.gif>	*Il file foto1.gif è nella cartella immagini che si trova nella directory corrente.*

Gli attributi di **img** sono:

- **align**, che determina la disposizione dell'immagine rispetto al testo che la circonda. Può assumere i valori:
  - **bottom**: il lato inferiore dell'immagine appare allineato alla base della riga di testo;
  - **middle**: il punto mediano dell'altezza dell'immagine si allinea alla base della riga di testo; il testo si spezza e prosegue sotto l'immagine;
  - **top**: il lato superiore dell'immagine appare allineato alla parte superiore della riga di testo;
  - **left**: l'immagine si posiziona sul lato sinistro della pagina e il testo scorre intorno a lei sul lato destro;
  - **right**: l'immagine si posiziona sul lato destro della pagina e il testo scorre intorno a lei sul lato sinistro.
- **width**, che determina la larghezza dell'immagine. La dimensione può essere specificata in pixel o in percentuale.

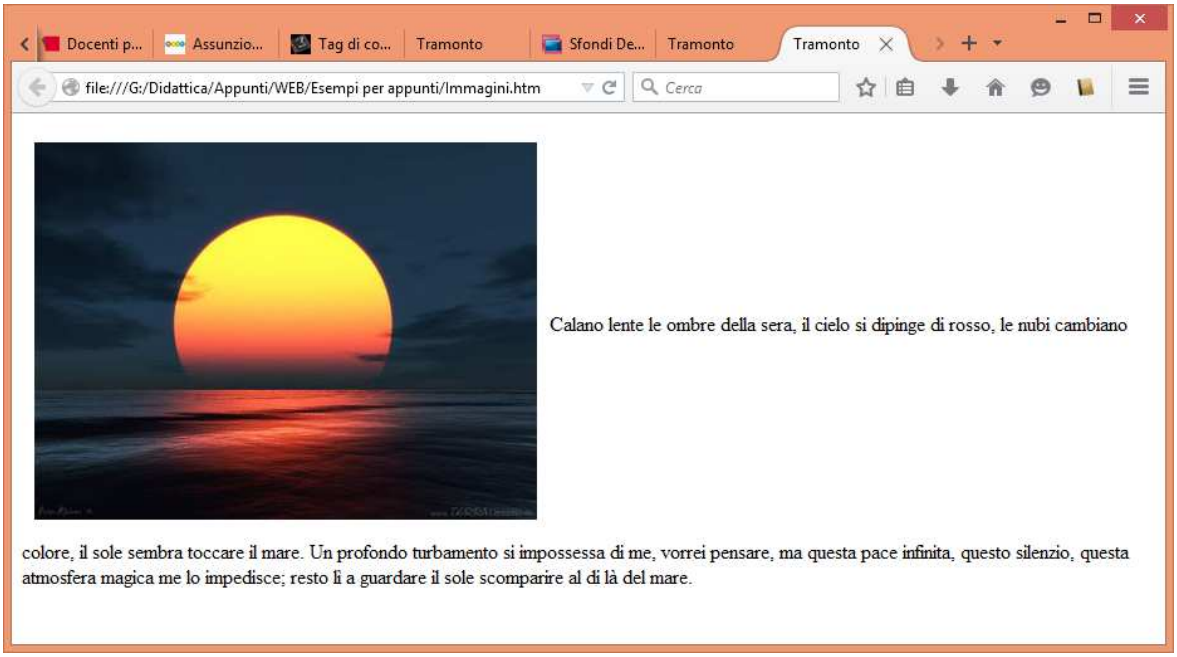
- **height**, che determina l'altezza dell'immagine. La dimensione può essere specificata in pixel o in percentuale.
- **alt**, che consente di inserire una descrizione dell'immagine, che verrà mostrata durante il caricamento della stessa o qualora l'immagine non risultasse reperibile.
- **title**, che consente di far apparire una tooltip quando il mouse si soffermerà sopra l'immagine.
- **border**, che consente di circondare l'immagine con un bordo di spessore opportuno.
- **hspace** e **vspace**, che consentono di determinare la distanza in pixel fra l'immagine e il testo, rispettivamente in orizzontale e in verticale.

Esempio:

```
<img src="tramonto.jpg" width="50%" height="50%" alt="immagine del tramonto"
                title="immagine del tramonto" border="2" hspace="10" vspace="15">
```

Esempio : Il seguente codice produce la pagina di seguito.

```
<!doctype HTML>
<html>
  <head>
      <title>Tramonto</title>
  </head>
  <body>
      <img src="tramonto.jpg" align="middle" height="300" width="400" hspace="10"
       vspace="15">Calano lente le ombre della sera, il cielo si dipinge di rosso, le nubi
      cambiano colore, il sole sembra toccare il mare. Un profondo turbamento si impossessa di
      me, vorrei pensare, ma questa pace infinita, questo silenzio, questa atmosfera magica me
      lo impedisce;  resto l&igrave; a guardare il sole scomparire al di l&agrave; del mare.
  </body>
</html>
```



Le linee orizzontali

Per meglio organizzare il testo all'interno di una pagina, è possibile inserire righe di separazione che dividano orizzontalmente le parti del discorso. Il tag per inserire una riga è <hr>.
Gli attributi sono:

- **size** - determina lo spessore della linea, espresso in pixel attraverso un valore numerico che può variare da 1 a 100;
- **width** - determina la lunghezza in pixel o in percentuale della linea (100% = tutta la larghezza dello schermo);
- **align** - imposta la collocazione della linea e ammette i valori "**right**", "**left**" e "**center**";
- **color** - colora la linea.

Esempio: *<hr size="4" width="50%" align="center" color="green">*

Sul web sono disponibili anche molte linee in formato grafico, sia fisso che animato. In questo caso il loro inserimento in una pagina web avviene usando il tag <img>.

Esempio:

Le tabelle

Le tabelle sono usate per disporre i dati in modo ordinato per righe e colonne. Prima dei fogli di stile, sono state ache lrgamente utilizzate per la formattazione delle pagine, consentendo di disporre i contenuti di una pagina in modo più articolato. Una tabella inizia con il tag **<table>** e termina con il tag **</table>**. Ogni riga ha inizio con il tag **<tr>** e termina con il tag **</tr>**. In una riga ogni cella è racchiusa tra il tag **<td>** e **</td>**. Si possono usare i tag **<th>** e **</th>** anziché <td> e </td> per celle che hanno funzione d'intestazione; il testo in esse contenuto apparirà in grassetto in modo automatico.

La sintassi è la seguente:

```
<table>
    <tr>
      <td> … </td>
      <td> … </td>
       …
    </tr>
    <tr>
     <td> … </td>
     <td> … </td>
       …
    </tr>
```

Tra i tag <td> e </td> possono essere posizionati contenuti di qualunque tipo, sia testi che immagini.

...
</table>

Gli attributi di **table** sono:

- **align** – definisce l'allineamento della tabella; può assumere i valori **left**, **center** e **right**.
- **border** - definisce lo spessore di tutti i bordi della tabella; per default, vale 0, cioè il bordo non è visibile;
- **cellpadding** - definisce lo spazio fra il bordo della tabella e il contenuto della cella (d. 1 pixel)
- **cellspacing** - definisce la distanza fra una cella e l'altra; il valore predefinito è 2 pixel.
- **width** - definisce la larghezza della tabella (in valore assoluto o in percentuale);
- **height** - definisce l'altezza della tabella (in valore assoluto o in percentuale);
- **bgcolor** - definisce il colore dello sfondo della tabella;
- **bordercolor** – definisce il colore di tutti i bordi della tabella;
- **background** – definisce un'immagine di sfondo.

Gli attributi di <tr> sono:

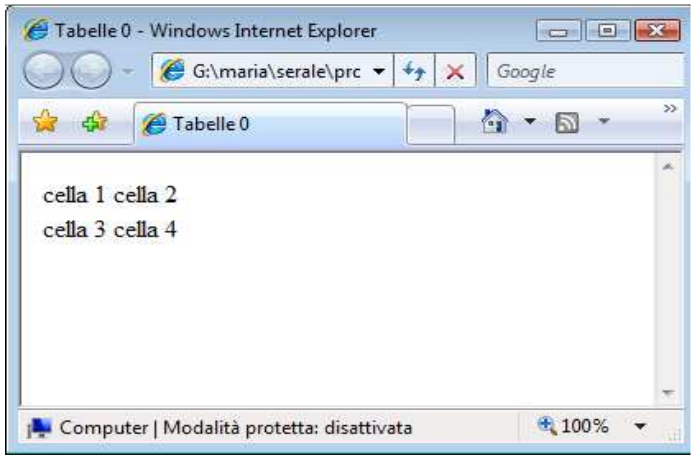
- **align** - definisce l'allineamento orizzontale di tutte le celle presenti nella riga; i valori possibili sono **left**, **center** e **right**;
- **valign** - definisce l'allineamento verticale di tutte le celle presenti nella riga; i valori possibili sono **top**, **middle** e **bottom**;
- **bordercolor** - definisce il colore dei bordi delle celle della riga;
- **height** - definisce l'altezza della riga (in valore assoluto o in percentuale);

Gli attributi di <td> sono:

- **align** - definisce l'allineamento orizzontale della cella; i valori possibili sono **left**, **center** e **right**;
- **valign** - definisce l'allineamento verticale della cella; i valori possibili sono **top**, **middle** e **bottom**;
- **width** - definisce la larghezza della singola cella;
- **height** - definisce l'altezza della cella (in valore assoluto o in percentuale);
- **bgcolor** - definisce il colore dello sfondo della singola cella;
- **bordercolor** - definisce il colore del bordo della singola cella;
- **rowspan** - fa sì che una cella risulti alta *n* celle, dove *n* è il valore attribuito a rowspan;
- **colspan** - fa sì che una cella risulti larga *n* celle, dove *n* è il valore attribuito a colspan.
- **background** – definisce un'immagine di sfondo

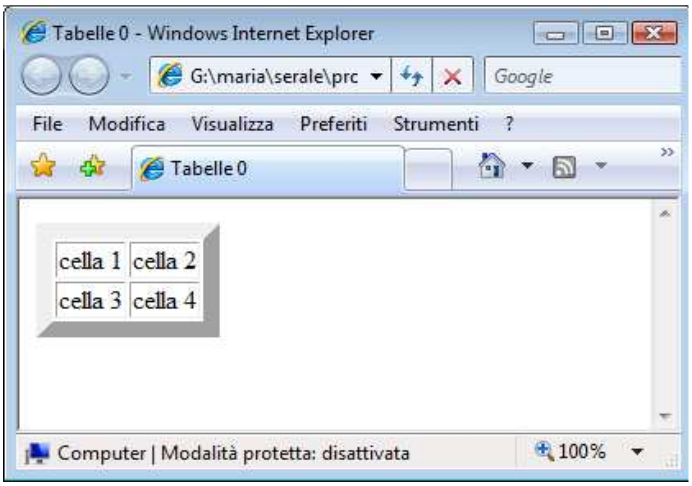
Esempio: tabella con border="0" (non specificato)

```
<table>
 <tr>
   <td>cella 1</td>
   <td>cella 2</td>
 </tr>
 <tr>
   <td>cella 3</td>
   <td>cella 4</td>
 </tr>
</table>
```



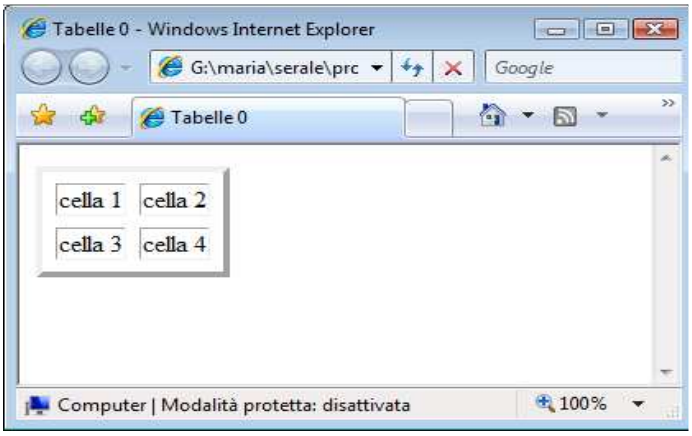
Esempio: tabella con border=”10”

```
<table border=”10”>
<tr>
  <td>cella 1</td>
  <td>cella 2</td>
 </tr>
 <tr>
  <td>cella 3</td>
  <td>cella 4</td>
 </tr>
</table>
```



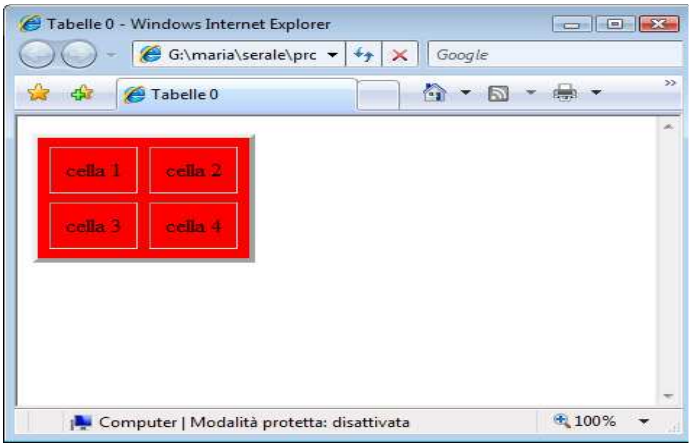
Esempio: tabella con cellspacing

```
<table border="4" cellspacing="8">
  <tr>
    <td>cella 1</td>
    <td>cella 2</td>
  </tr>
  <tr>
    <td>cella 3</td>
    <td>cella 4</td>
  </tr>
</table>
```



Esempio: cellpadding e bgcolor

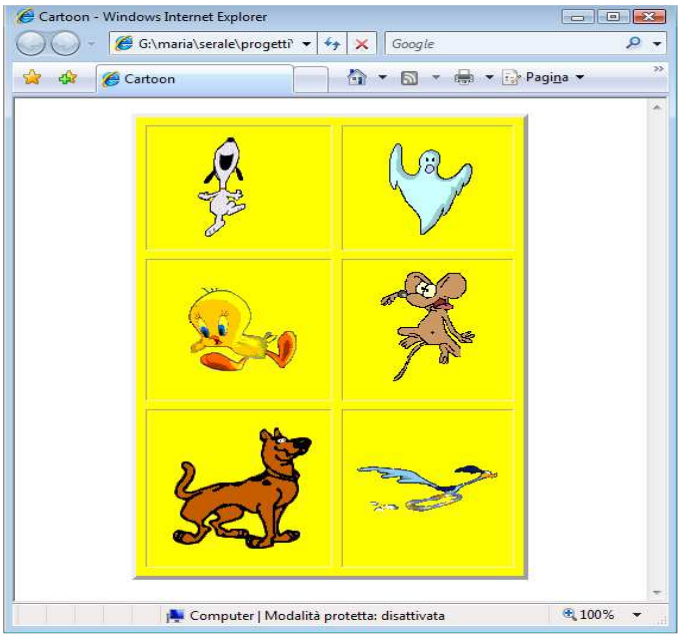
```
<table border="4" cellspacing="8"
cellpadding="10" bgcolor=”red”>
  <tr>
     <td>cella 1</td>
     <td>cella 2</td>
  </tr>
  <tr>
     <td>cella 3</td>
     <td>cella 4</td>
  </tr>
</table>
```



Esempio: immagini in una tabella

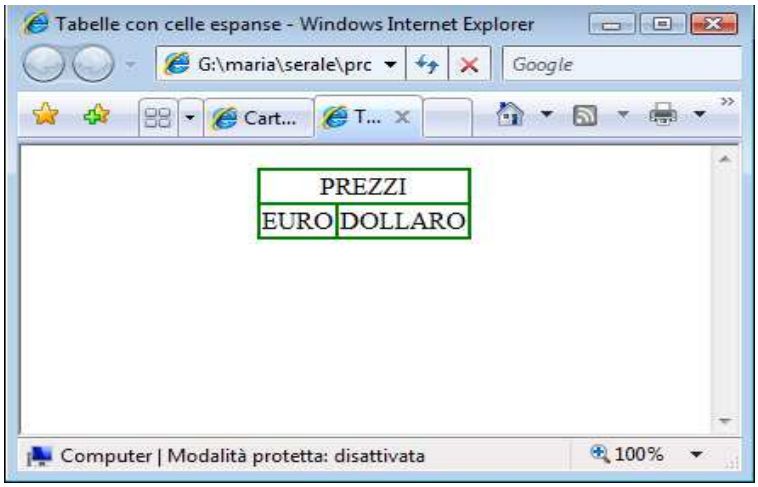
```
<html>
  <head>
    <title>Cartoon</title>
  </head>
  <body>
    <table align="center" border="4"
    cellspacing="8" cellpadding="10"
    bgcolor="yellow">
      <tr align="center">
        <td>
```

```
        <img src="./immagini/snoopy.gif"
      </td>
      <td>
        <img src="./immagini/spettro.gif">
      </td>
    </tr>
    <tr align="center">
      <td>
        <img src="./immagini/titty.gif">
      </td>
      <td>
        <img src="./immagini/topo.gif">
      </td>
    </tr>
    <tr align="center">
      <td>
        <img src="./immagini/scooby.gif">
      </td>
      <td>
        <img src="./immagini/bibi.gif">
      </td>
    </tr>
  </table>
 </body>
</html>
```



Tutte le celle di una colonna devono avere la stessa larghezza e tutte le celle di una riga devono avere la stessa altezza. Se sono indicati valori differenti, il browser adotta quello più elevato. Per creare una tabella con un numero non costante di celle per ogni riga/colonna, si usano le **celle espanse**, realizzate attraverso gli attributi **rowspan** e **colspan** del tag <td> o <th>.

Esempio:




```
<html>
 <head>
  <title>Tabelle con celle espanse</title>
 </head>
 <body>
  <table width="30%" border="1" bordercolor="green" cellspacing="0">
    <tr>
      <td colspan="2" align="center">
          PREZZI
      </td>
    </tr>
    <tr>
```

```
        <td width="50%" align="center">EURO</td>
        <td align="center">DOLLARO</td>
    </tr>
    </table>
  </body>
</html>
```

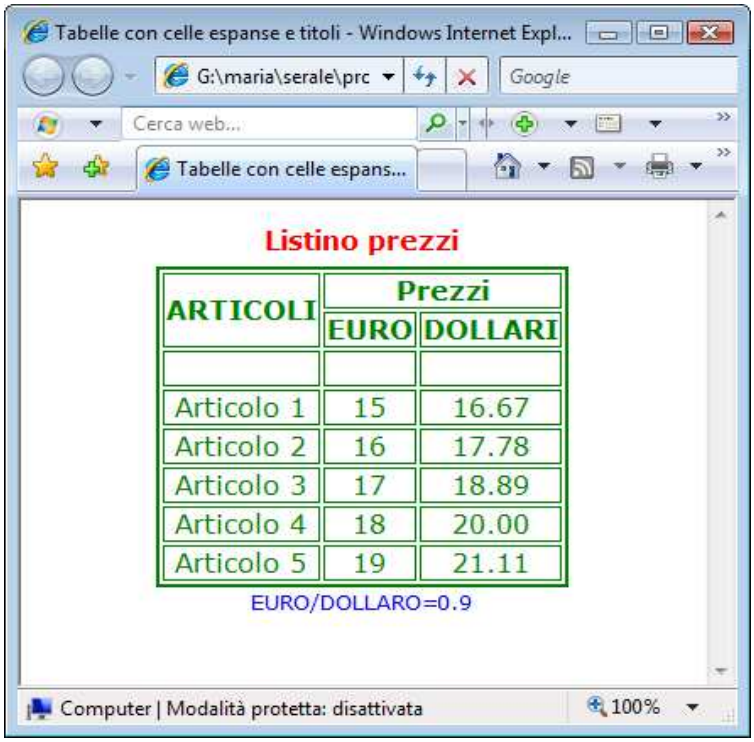
Un altro tag utile alle strutture delle tabelle è **<caption>**, che serve per aggiungere titoli alla tabella. Il tag <caption>, seguito dal titolo e dall'obbligatorio tag di chiusura </caption> si collocano subito dopo il tag <table>. Attributo di <caption> è **align**, che può assumere i seguenti valori:

- **top**, per far apparire il titolo prima della tabella;
- **bottom**, per far apparire il titolo alla fine della tabella.

Esempio:

```
<html>
<head>
<title>Tabelle con celle espanse e titoli</title>
</head>
<body text="green">
<font face="verdana">
<table width="30%" border="2" bordercolor="green" align="center">
        <caption align="top">
        <b><font color="red">Listino prezzi</font></b>
        </caption>
        <caption align="bottom">
        <font color="blue" size="2">EURO/DOLLARO=0.9</font>
        </caption>
        <tr>
                <th rowspan="2" align="center">ARTICOLI</th>
                <th align="center" colspan="2">Prezzi</th>
        </tr>
        <tr align="center">
                <th>EURO</th>
                <th>DOLLARI</th>
        </tr>
        <tr>
                <td> </td>
                <td> </td>
                <td> </td>
        </tr>
        <tr align="center">
                <td>Articolo 1</td>
                <td>15</td>
                <td>16.67</td>
        </tr>
        <tr align="center">
                <td>Articolo 2</td>
                <td>16</td>
                <td>17.78</td>
        </tr>
        <tr align="center">
                <td>Articolo 3</td>
                <td>17</td>
                <td>18.89</td>
        </tr>
        <tr align="center">
                <td>Articolo 4</td>
```

```
            <td>18</td>
            <td>20.00</td>
        </tr>
        <tr align="center">
            <td>Articolo 5</td>
            <td>19</td>
            <td>21.11</td>
        </tr>
</table>
</font>
</body>
</html>
```



La riga

```
        <tr>
            <td> </td>
            <td> </td>
            <td> </td>
        </tr>
```

inserisce spazi nelle celle vuote, per evitare che i bordi collassino su se stessi.

I link e l'ipertestualità

Una delle caratteristiche che ha fatto la fortuna del web è l'essere costituito non da **testi** ma da **ipertesti**. Il collegamento, o **link**, tra un documento e l'altro è una parola o una frase o anche un'immagine, su cui si deve cliccare per "saltare" alla parte di testo collegata, che può essere un'altra pagina (sullo stesso server o su un server diverso), oppure un altro punto della pagina stessa (collegamento interno).

Link che puntano ad altri documenti

Per creare un link ad un altro documento si usa il tag **<a>** con l'attributo **href**, seguito dal percorso del documento a cui punta il link, che sarà racchiuso tra i tag <a> e .
Il seguente esempio

Per ulteriori ****informazioni**** sul funzionamento dei link clicca sulla parte evidenziata.

dà come risultato:

Per ulteriori informazioni sul funzionamento dei link clicca sulla parte evidenziata

Il link è la parola informazioni, che appare in blu e sottolineata. Cliccando sul link, il browser visualizza la pagina di nome "esempi.html", che, in questo caso, ha lo stesso pathname della pagina da cui è partito il collegamento, altrimenti occorre specificarne il percorso completo.

Un altro attributo del tag <a> è title, che ha come valore una stringa di caratteri, ad esempio una descrizione del link, che sarà visualizzata in una tooltip quando il mouse passerà sul link.
Nell'esempio precedente:

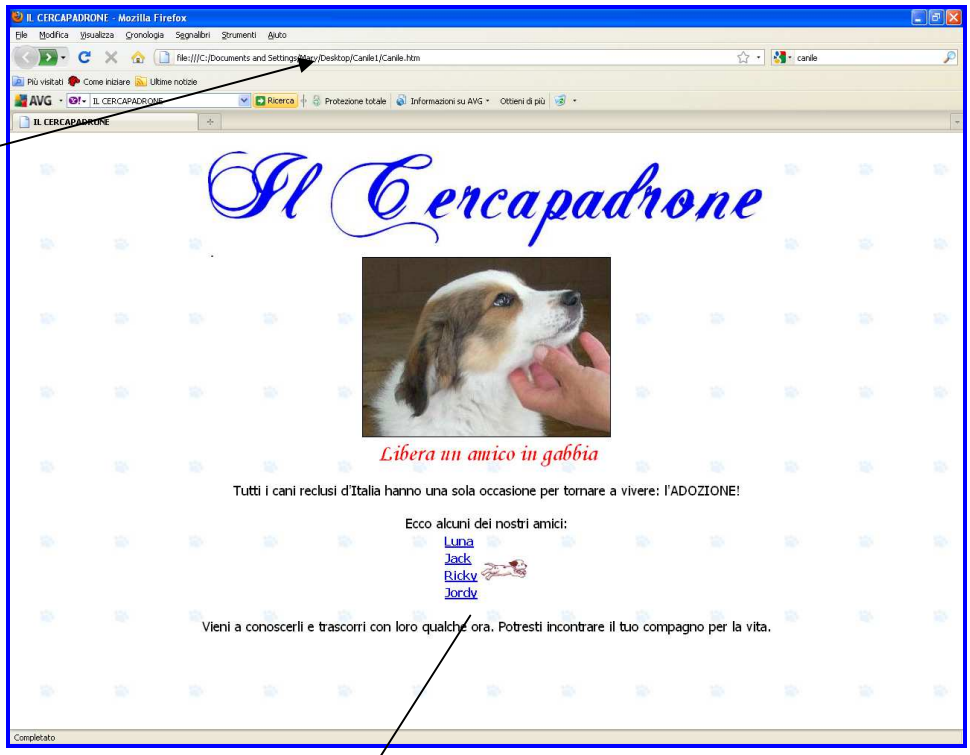
Per ulteriori ****informazioni**** sul funzionamento dei link clicca sulla parte evidenziata.

Esempio:

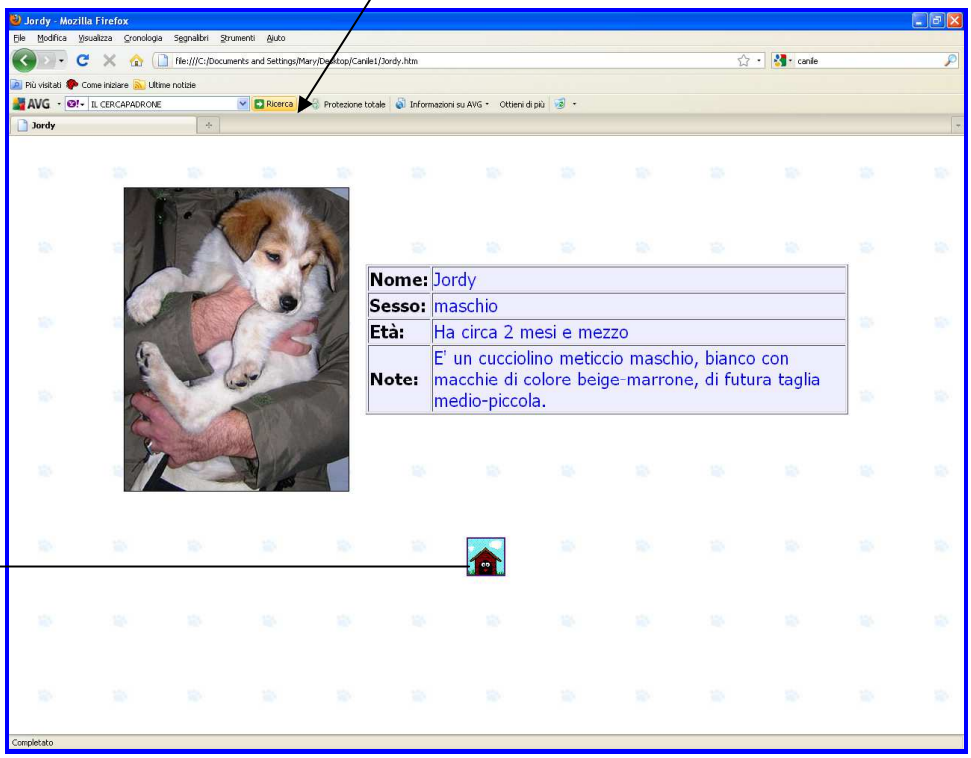
```
<html>
<head>
        <title>IL CERCAPADRONE</title>
</head>
<body background="./immagini/sfondoimpronta.jpg">
<div align="center">
<img src="./immagini/titolo.jpg"><br>
<img src="./immagini/amicocane.jpg" border="1"><br>
<font face="monotype corsiva" color="red" size="6">
Libera un amico in gabbia
</font><br><br>
<font face="Tahoma" size="4">
Tutti i cani reclusi d'Italia hanno una sola occasione per tornare a vivere: l'ADOZIONE!<br><br>
Ecco alcuni dei nostri amici:
<table>
        <tr>
          <td><b><a href="Luna.htm">Luna</a></b></td>
          <td rowspan="4" align="center" valign="middle">
                <img src="./immagini/dalmatacucciolocorre.gif" align="bottom" width="64" height="40">
          </td>
        </tr>
        <tr><td><b><a href="Jack.htm">Jack</a></b></td></tr>
        <tr><td><b><a href="Ricky.htm">Ricky</a></b></td></tr>
        <tr><td><b><a href="Jordy.htm">Jordy</a></b></td></tr>
</table>
<br>
```

Vieni a conoscerli e trascorri con loro qualche ora. Potresti incontrare il tuo compagno per la vita.
**
* *
</body>
</html>

Viene visualizzata la pagina seguente, dove sono presenti quattro link. Cliccando su ciascuno di essi, ad esempio sull'ultimo, il browser mostra una nuova pagina:



Canile.htm



Jordy.htm

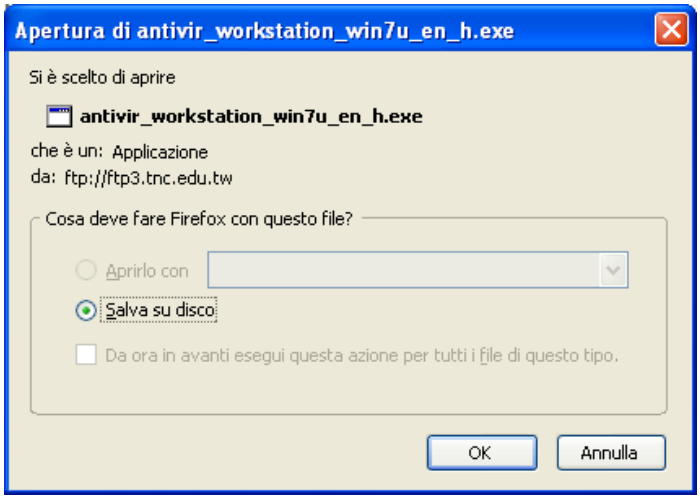
Nella pagina Jordy.htm, che è richiamata dalla pagina Canile.htm, c'è un link di ritorno alla pagina principale (home page), realizzato attraverso una gif animata. La gif appare con un bordo azzurro.
Il codice è il seguente:

```
<html>
<head>
<title>Jordy</title>
</head>
<body background="./immagini/sfondoimpronta.jpg"><br><br>
<table  width="80%" align="center" cellspacing="20">
   <tr>
      <td><img src="./immagini/Jordy.jpg" border="1"></td>
       <td valign="middle">
            <table border="1"  width="100%" valign="middle" bgcolor="#eeeeff">
                  <tr>
                        <td><font size="5" face="Tahoma" color="black">
                              <b> Nome:</b></font>
                        </td>
                        <td><font size="5" face="Tahoma" color="blue">Jordy</font>
                        </td>
                  </tr>
                  <tr>
                        <td><font size="5" face="Tahoma" color="black">
                              <b>Sesso:</b></font>
                        </td>
                        <td><font size="5" face="Tahoma" color="blue">maschio</font>
                        </td>
                  </tr>
                  <tr>
                        <td><font size="5" face="Tahoma" color="black">
                                    <b>Età:</b></font>
                        </td>
                        <td><font size="5" face="Tahoma" color="blue">
                              Ha circa 2 mesi e mezzo</font>
                        </td>
                  </tr>
                  <tr>
                        <td><font size="5" face="Tahoma" color="black">
                              <b>Note:</b></font>
                        </td>
                        <td><font size="5" face="Tahoma" color="blue">
                              E' un cucciolino meticcio maschio, bianco con macchie di
                              colore beige-marrone, di futura taglia medio-piccola.</font>
                        </td>
                  </tr>
                  </table>
      </td>
   </tr>
</table>
<br>
<br>
<div align="center"><a href="Canile.htm"><img src="./immagini/Home.gif"></a></div>
</body>
</html>
```

È possibile specificare in quale finestra la pagina linkata deve essere aperta: di default la pagina viene aperta all'interno del documento stesso, ma è possibile specificare che la pagina sia aperta in una nuova finestra, attraverso l'attributo **target** con valore "**_blank**".

La destinazione di un link può essere una pagina HTML ma anche una qualsiasi altra risorsa, come un'immagine, un file pdf, un file zip, o un file exe. Il browser si comporterà in modo differente a seconda della risorsa:

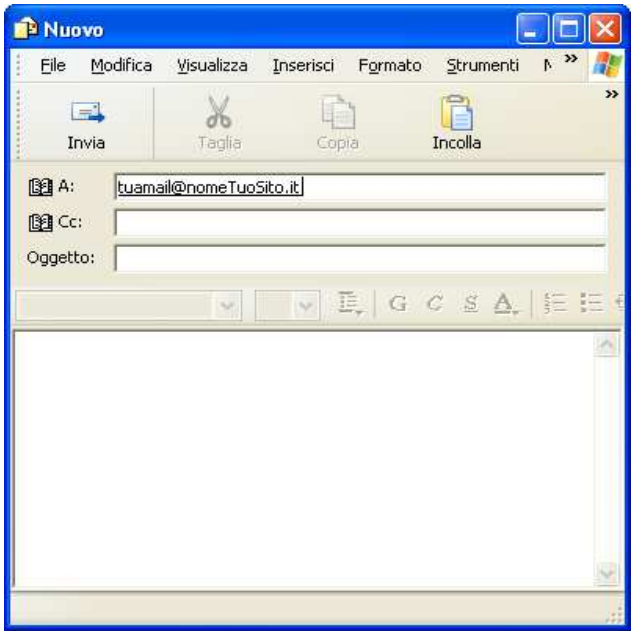
- per i file .htm, il browser apre la pagina.
- per i file d'immagine (.gif, .jpg, .png), il browser visualizza l'immagine.
- per i file contenenti documenti .doc e .pdf, il browser apre il documento, ma solo se l'utente ha installato sul proprio PC l'apposito **plugin**. Con questo termine si identifica un programma "aggiuntivo" che aggiunge nuove funzionalità al browser; se non è installato il plugin, il sistema chiederà all'utente se salvare il file.
- per i file .zip, .com e .exe, viene generalmente chiesto all'utente di scaricare il file, in quanto, per motivi di sicurezza, non è opportuno eseguire un file eseguibile direttamente dal web. Firefox visualizza, ad esempio, la finestra a fianco.



Il link può anche trasmettere il collegamento ad un indirizzo e-mail. In questo caso si aprirà direttamente il client di posta dell'utente con l'indirizzo e-mail pre-impostato. La sintassi è:

```
<a href="mailto:tuaMail@nomeTuoSito.it">Mandami una email</a>.
```

Che dà come risultato il link: Mandami una e-mail.
Se il client di posta elettronica pre-impostato è Outlook, cliccando sul link apparirà:



Si può infine far riferimento ad una pagina di un altro sito web. Ad esempio:

Leggi le risorse sui fogli di stile

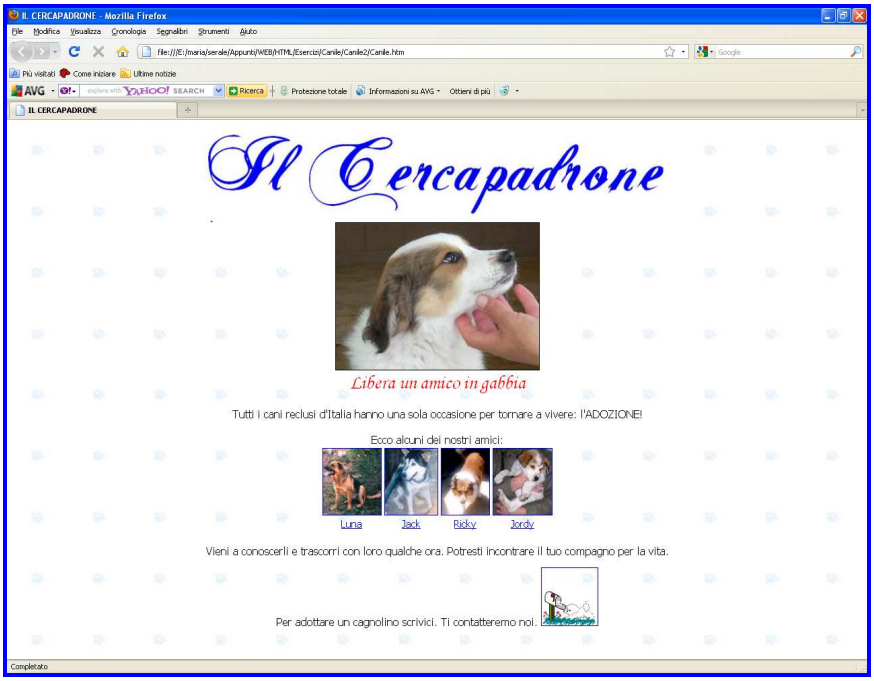
Il link fa riferimento ad una particolare directory di un sito web:

http://	Indica al browser di utilizzare il protocollo per navigare nel web (l'http)
www.html.it/	Indica di fare riferimento al sito www.html.it
css/	Indica che la risorsa indicata si trova all'interno della cartella "css"
index.html	Indica che il file da collegare è quello il cui nome è "index.html"

l'indirizzo di una pagina web prende il nome di **URL** (Uniform Resource Locator**)**

Esempio

Vediamo una seconda versione dell'home page dell'esempio precedente:



Questa volta i link alle schede dei cani sono doppi: il nome e una miniatura dell'immagine, che prende il nome di **thumbnail** (unghia del pollice). Inoltre alla fine della pagina c'è una gif animata che collega ad una casella di posta elettronica. Il codice della pagina è il seguente:

```
<html>
<head>
        <title>IL CERCAPADRONE</title>
</head>
<body background="./immagini/sfondoimpronta.jpg">
<div align="center">
<img src="./immagini/titolo.jpg"><br>
<img src="./immagini/amicocane.jpg" border="1"><br>
<font face="monotype corsiva" color="red" size="6">Libera un amico in
gabbia</font><br><br>
<font face="Tahoma" size="4">
Tutti i cani reclusi d'Italia hanno una sola occasione per tornare a vivere: l'ADOZIONE!<br><br>
Ecco alcuni dei nostri amici:
<table align="center">
  <tr align="center">
        <td><a href="Luna.htm"><img src="./immagini/LunaSmall.jpg"></a></td>
        <td><a href="Jack.htm"><img src="./immagini/JackSmall.jpg"></a></td>
```

```
        <td><a href="Ricky.htm"><img src="./immagini/RickySmall.jpg"></a></td>
        <td><a href="Jordy.htm"><img src="./immagini/JordySmall.jpg"></a></td>
  </tr>
  <tr align="center">
        <td><a href="Luna.htm">Luna</a></td>
        <td><a href="Jack.htm">Jack</a></td>
        <td><a href="Ricky.htm">Ricky</a></td>
        <td><a href="Jordy.htm">Jordy</a></td>
  </tr>
</table>
<br>
Vieni a conoscerli e trascorri con loro qualche ora. Potresti incontrare il tuo compagno per la vita.
<br>
<p align="center">Per adottare un cagnolino scrivici. Ti contatteremo noi.
<a href="mailto:BelMusino@WebPlace.it"><img src="./immagini/scrivici.gif"></a>
<br>
</font>
</div>
</body>
</html>
```

I link interni (ancore)

E' anche possibile impostare link interni, ossia effettuare un collegamento ad una determinata posizione dello stesso documento html.
A tale scopo è necessario "marcare" il punto (o i punti) nella pagina a cui saltare, usando la sintassi:

…

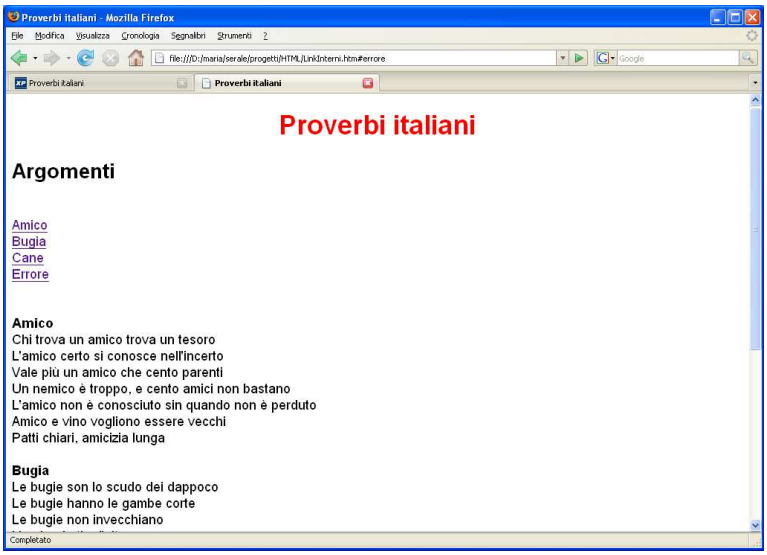
dove i puntini di sospensione indicano la parte di testo da individuare con il nome *posizione1*.
Quando da un qualsiasi altro punto nella pagina si vuole saltare a questo, si deve usare la sintassi:

link alla posizione

Se *posizione1* è all'interno di un'altra pagina, si scrive invece:

link alla posizione

Esempio:



```
<html>
<head><title>Proverbi italiani</title></head>
<body>
<font face="Arial" size="4">
<h1 align="center">
<font color="red">
 Proverbi italiani
</font></h1>
<h2>Argomenti</h2>
<br>
<a href="#amico">Amico</a><br>
<a href="#bugia">Bugia</a><br>
<a href="#cane">Cane</a><br>
<a href="#errore">Errore</a>
<br><br><br>
<a name="amico"><b>Amico</b><br>
Chi trova un amico trova un tesoro<br>
L'amico certo si conosce nell'incerto<br>
Vale più un amico che cento parenti<br>
Un nemico è troppo, e cento amici non bastano<br>
L'amico non è conosciuto sin quando non è perduto<br>
Amico e vino vogliono essere vecchi<br>
Patti chiari, amicizia lunga<br>
</a><br>
<a name="bugia"><b>Bugia</b><br>
Le bugie son lo scudo dei dappoco<br>
Le bugie hanno le gambe corte<br>
Le bugie non invecchiano<br>
Una bugia tira l'altra<br>
La bugia è come una valanga, più rotola più s'ingrossa<br>
</a><br>
<a name="cane"><b>Cane</b><br>
Can che abbaia non morde<br>
Cane vecchio non abbaia invano<br>
Cane non mangia cane<br>
Guardati da can rabbioso e da uomo sospettoso<br>
Il cane rode l'osso perché non lo può inghiottire<br>
</a><br>
<a name="errore"><b>Errore</b><br>
Gli errori de' medici son ricoperti dalla terra, quelli dei ricchi dai danari<br>
Errare è umano, perseverare è diabolico<br>
Non c'è uomo che non erri, né cavallo che non sferri<br>
Chi tenta di scusare un errore, erra un'altra volta<br>
Sbaglia il prete all'altare e il contadino all'aratro<br>
Non tutte le ciambelle riescono col buco<br>
Chi favella, erra<br>
</a>
</font>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><
br><br><br>
</body>
</html>
```

Cliccando su ciascuno dei link a inizio pagina, il browser visualizza la parte di testo corrispondente.
La penultima riga del codice presenta una serie di tag <br>, il cui scopo è quello di prolungare la pagina, onde evitare che, in seguito al link interno, una sezione non appaia a inizio pagina. Eliminare la riga per comprenderne l’effetto.

Mappe d'immagine

E' possibile suddividere un'immagine in diverse aree (creando una mappa, da cui appunto il nome) ed associare ad ognuna di queste aree la possibilità di richiamare un link diverso.
In pratica, una sola immagine è in grado di richiamare link diversi a seconda del punto in cui viene cliccata.

Per indicare che un'immagine è una mappa d'immagine, si usa l'attributo **usemap** del tag **<img>**:

<**img src=**"nome_file_immagine" **usemap=**"#nome_mappa" >

Il valore di **usemap** è introdotto dal simbolo **#** ed è il nome della mappa, che viene definito attraverso un altro tag, il tag **<map>**. Tra **<map>** e **</map>** sono indicati, per ogni area della mappa, il tipo di forma dell'area, le coordinate dei punti che la individuano, il link associato all'area e un'eventuale tooltip. Ogni area è contenuta tra il tag **<area>** e la sua chiusura **</area>**.

```
<map name="nome_mappa">
        <area shape="…" coords="……" href="pagina_area1.htm"  title= "…">
        <area shape="…" coords="……" href=" pagina_area2.htm" title= "…">
        <area shape="…" coords="……" href=" pagina_area3.htm" title= "…">
        <area shape="…" coords="……" nohref
        ….
</map>
```

L'attributo **name** di <map> specifica il nome della mappa in modo che usemap possa fare riferimento ad essa.

L'attributo **shape** (forma) del tag <area> fornisce la forma dell'area specificata; **rect** sta per rettangolo, che è anche il valore di default. Altre forme possibili sono **circle** (cerchio) e **poly** (poligono), ma solo rect è riconosciuto sicuramente da tutti i browser.

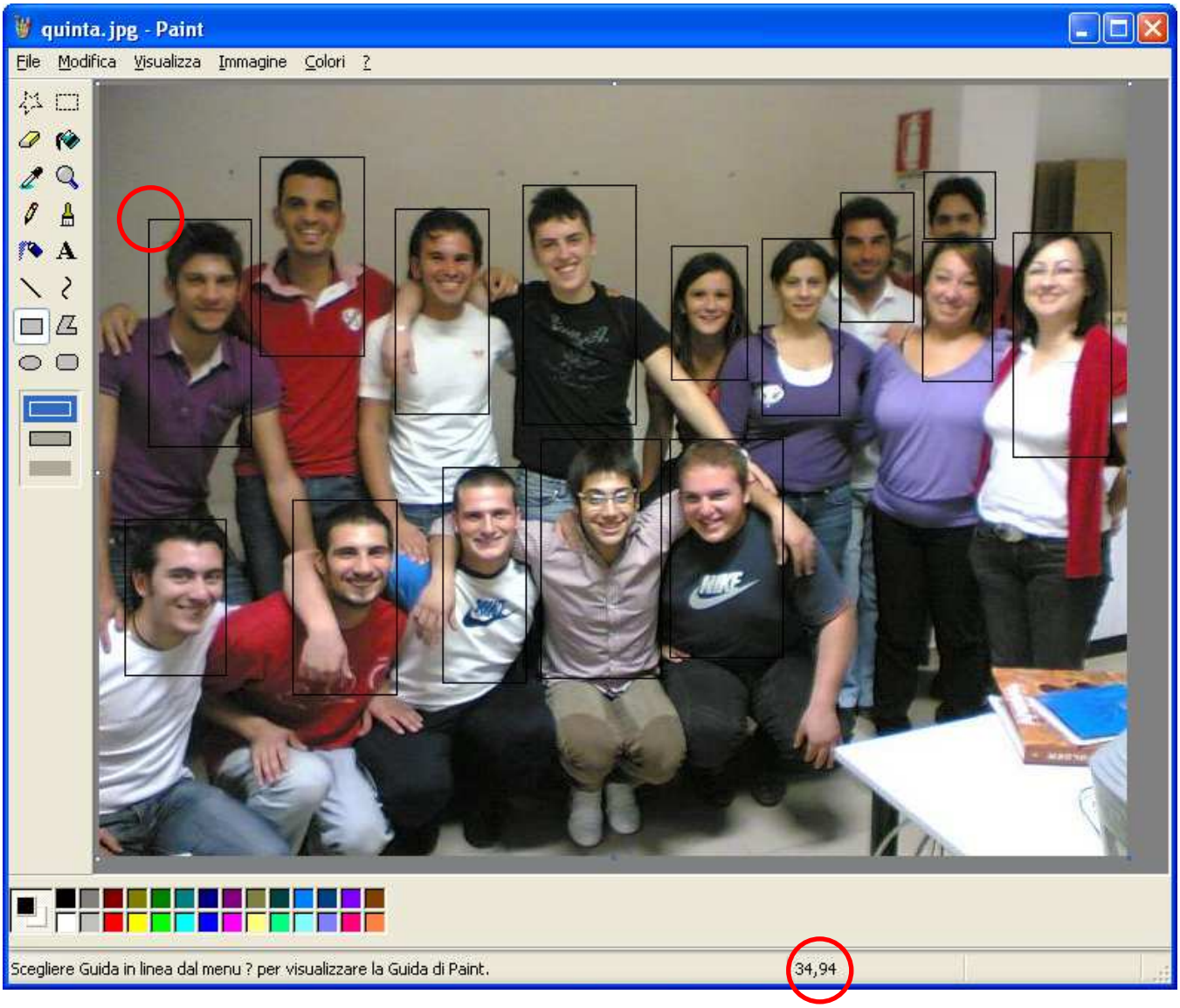
L'attributo **coords** del tag <area> fornisce le coordinate che individuano la forma, utilizzando la numerazione dei pixel dell'immagine, che sono disposti su colonne e righe numerate a partire da 0; il punto (0,0) è in alto a sinistra.
Per un rettangolo le coordinate sono: colonna e riga del vertice in alto a sinistra, colonna e riga del vertice in basso a destra. Se, per esempio, si volesse specificare l'intera area di una immagine da 100 x 100 pixel, le coordinate sarebbero:"0,0,99,99".
Per il cerchio si specificano le coordinate dal centro e il raggio. Per il poligono si specificano le coordinate di tutti vertici (colonna e riga di ciascuno).
Le coordinate dei punti possono essere facilmente determinate attraverso un programma di grafica. Alcune applicazioni sono in grado di generare direttamente il codice html della mappa.

L'attributo **href** del tag <area> richiama il link ad esso associato; scrivendo **nohref**, si indica che l'area non ha un link associato.

L'attributo **title** del tag <area> fa in modo che compaia una tooltip quando il mouse è su una determinata area dell'immagine.

Esempio:

Per creare la mappa dell'immagine seguente, basta aprire l'immagine in un programma di grafica come **Paint**, tracciare i rettangoli che delimitano le diverse aree di sensibilità e, muovendo il cursore sui vertici dei diversi rettangoli, rilevare le coordinate nella barra di stato, in basso a destra.

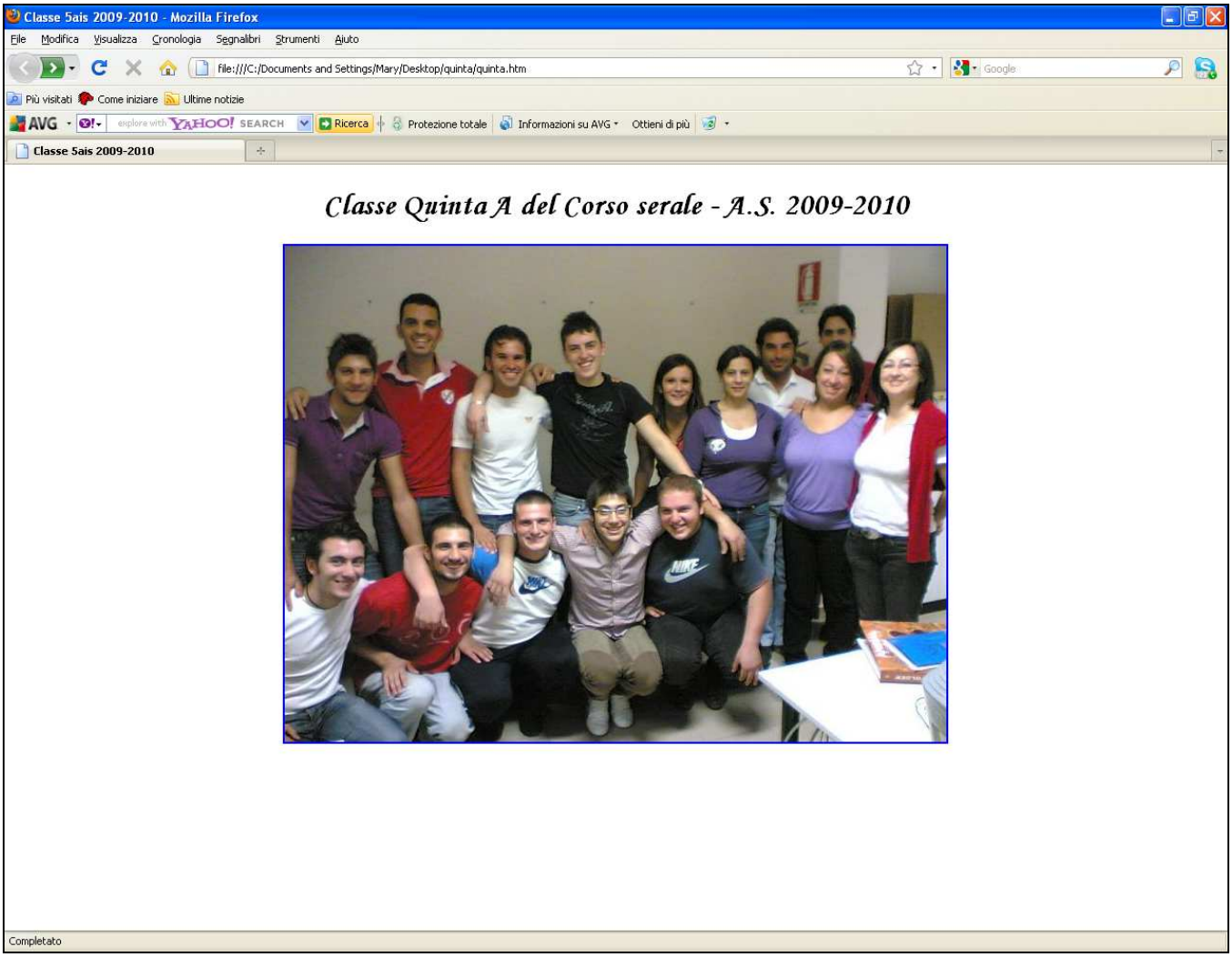


Il codice seguente

```
<html>
<head>
	<title>Classe 5ais 2009-2010</title>
</head>
<body>
	<font face="Monotype corsiva" color="blu">
	<h1 align="center">Classe Quinta A del Corso serale - A.S. 2009-2010</h1>
	<div align="center"><img src="quinta.jpg" usemap="#mappaquinta"></div>
	<map name="mappaquinta">
		<area shape="rect" coords="34, 94, 103, 242" href="torella.htm" title="Antonio Torella">
		<area shape="rect" coords="107, 50, 179, 181" href="russol.htm" title="Luigi Russo">
		<area shape="rect" coords="199, 83, 261, 222" href="caputo.htm" title="Giuseppe Caputo">
		<area shape="rect" coords="284, 66, 361, 227" href="roca.htm" title="Piergiorgio Roca">
		<area shape="rect" coords="385, 109, 436, 196" href="mennuni.htm" title="Rosa Mennuni">
		<area shape="rect" coords="446, 103, 499, 222" href="battista.htm" title="Carmela Battista">
```

```
            <area shape="rect" coords="499, 72, 548, 159" href="strafezza.htm"
            title="Gerardo Emauele Strafezza">
            <area shape="rect" coords="556, 58, 604, 104" href="zangrilli.htm"
            title="Antonio Zangrilli">
            <area shape="rect" coords="554, 107, 601, 200" href="dimicco.htm" title="Maria
            Dimicco">
            <area shape="rect" coords="614, 99, 681, 250" href="bruni.htm" title="Giovanna
            Bruni">
            <area shape="rect" coords="17, 291, 86, 396" href="russor.htm" title="Roberto
            Pio Russo">
            <area shape="rect" coords="130, 279, 202, 408" href="bove.htm" title="Stefano
            Bove">
            <area shape="rect" coords="231, 258, 287, 402" href="caldarisi.htm"
            title="Benito Caldarisi">
            <area shape="rect" coords="297, 238, 379, 398" href="zippone.htm"
            title="Savino Zippone">
            <area shape="rect" coords="384, 238, 460, 384" href="pagliaro.htm" title="Nicola
            Pagliaro">
        </map>
</body>
</html>
```

produce dunque la pagina



dove l'immagine di ogni studente è linkabile.

I Frame

I frame consentono una divisione della pagina in sezioni indipendenti tra loro, in modo da visualizzare più pagine dentro una sola.
Quando si crea una pagina con frame, occorre usare il tag **<frameset>** anziché **<body>** e specificare, tra **<frameset>** e **</frameset>**, la struttura dei frame usando il tag **<frame>** per ciascuna sezione della pagina. Si possono creare sia frame orizzontali che frame verticali. All'interno di un frame è inoltre possibile creare un altro framset, ma non si possono avere più frameset in cascata (uno di seguito all'altro, o in orizzontale o in verticale), perché un frameset occupa sempre l'intera pagina (o l'intero frame in cui è insertito).

Attributi di **<frameset>** sono:

- **border** - indica lo spessore del bordo per tutti i frame figli;
- **bordercolor** - indica il colore del bordo per tutti i frame figli;
- **cols** - definisce numero e dimensione dei frame verticali da creare; un numero senza percentuale indica la dimensione della colonna in pixel, con percentuale indica la larghezza del frame relativamente alla finestra del browser. Si possono specificare più colonne, racchiudendo tra doppi apici le dimensioni in un elenco separato da virgole; scrivendo un asterisco al posto di un numero, si indica che il frame corrispondente occupa tutto lo spazio rimanente.

  Esempio: <frameset cols="20%,"> equivale a <frameset cols="20%,80%">*
- **rows** - definisce il numero e la dimensione delle righe che si desidera creare; si usa come **cols**.
- **frameborder** - specifica se i frame sono visualizzati con un bordo. Il valore 1 indica la presenza del bordo, mentre 0 lo disabilita. I singoli frame possono ridefinire questo attributo.

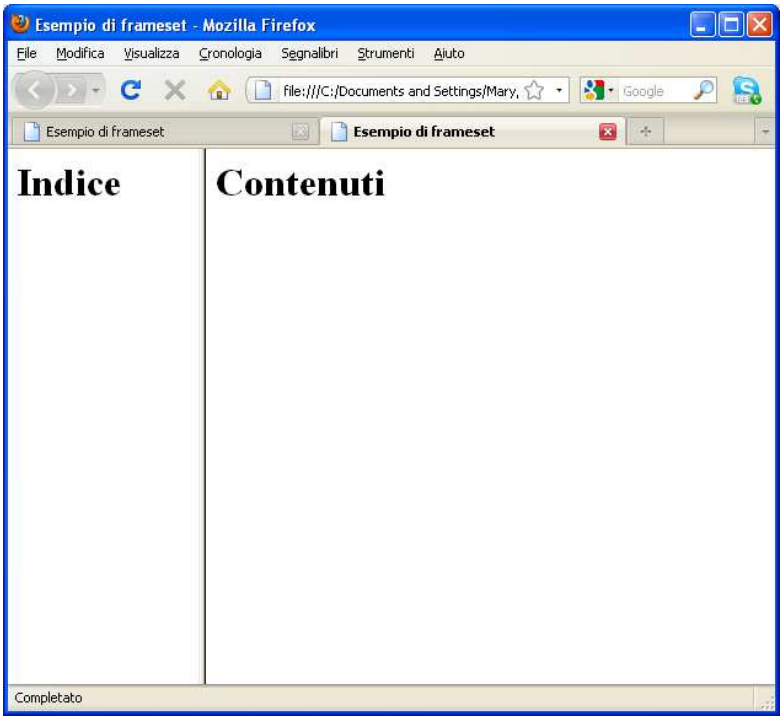
Attributi di **<frame>** sono:

- **frameborder** - specifica se il frame è visualizzato con un bordo. Il valore 1 ne indica la presenza, mentre 0 lo disabilita.
- **name** - indica il nome con cui far riferimento al frame.
- **noresize** - se presente, evita che l'utente possa ridimensionare il frame.
- **scrolling** - specifica se deve apparire una barra di scorrimento nel frame. I valori possibili sono **yes**, **no**, **auto**; se impostato su **auto**, che è l'impostazione di default, è il browser a determinare l'eventuale creazione della barra.
- **src** - indica il file da visualizzare all'interno del frame; se non si specifica un attributo src, lo spazio in cui dovrebbe apparire il frame risulterà vuoto.
- **marginheight** e **marginwidth** - permettono di impostare la distanza verticale (**marginheigth**) e orizzontale (**marginwidth**) tra i bordi del frame e il suo contenuto.

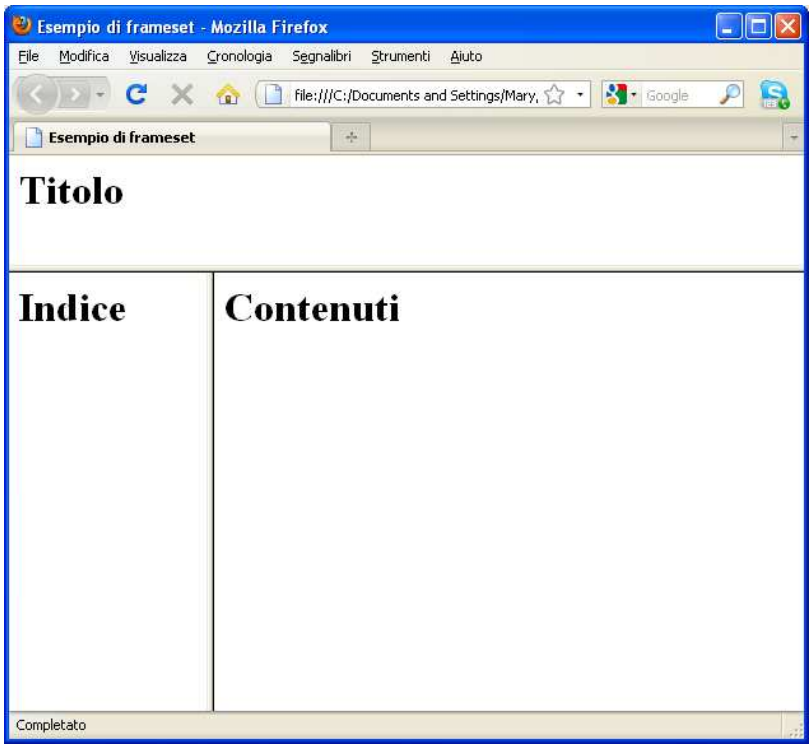
Esempio:

Pagina con due frame verticali. Si osservi che:
- ci sono <frameset> e </frameset> al posto di <body> e </body>,
- <frameset> imposta la larghezza delle colonne,
- in ogni frame viene visualizzata una pagina, il cui contenuto deve ovviamente essere adeguato alla larghezza del frame.

```
<html>
<head>
    <title>Esempio di frameset</title>
</head>
<frameset cols="25%,75%">
    <frame name="frameindice" src="indice.htm">
    <frame name="framecontenuti" src="contenuti.htm" >
</frameset>
</html>
```



Esempio:
Pagina con un frame orizzontale e due frame verticali.
Il frameset principale ha due righe, la seconda delle quali è a sua volta un frameset, questa volta con due colonne.
L'attributo noresize fa in modo che tutti i frame siano non ridimensionabili.

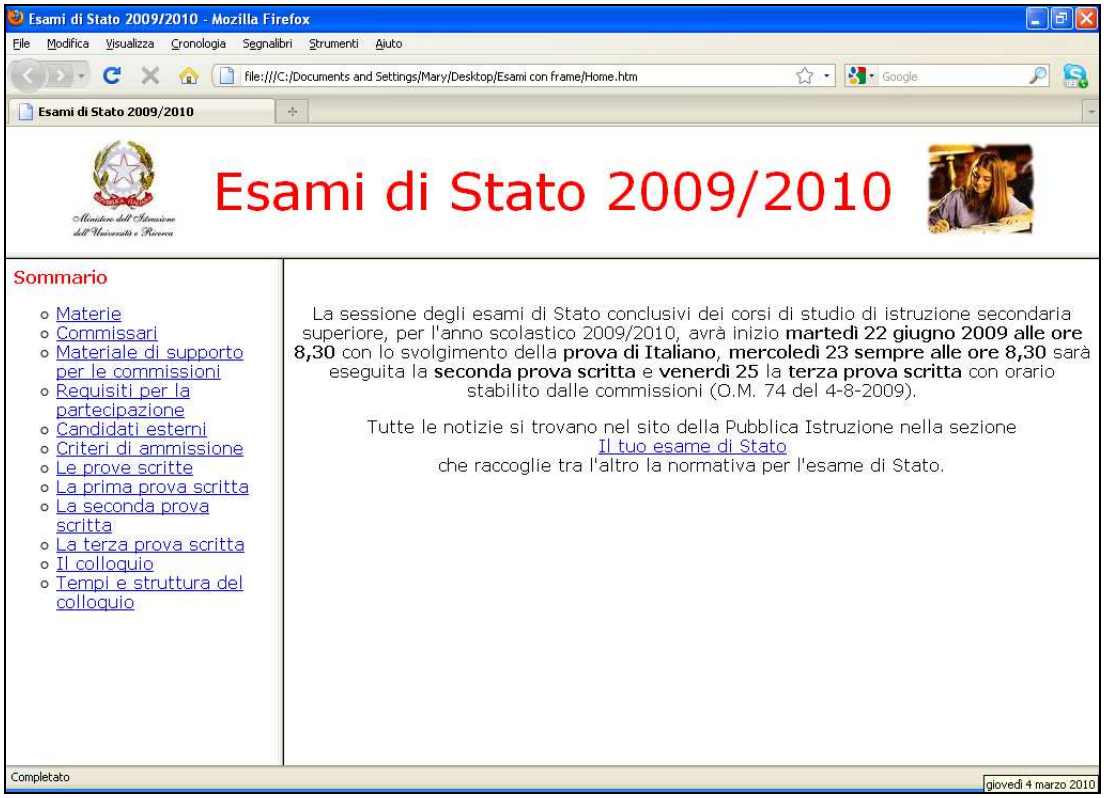



```
<html>

<head>
    <title>Esempio di frameset</title>
</head>
```

```
<frameset rows="20%,80%">

        <frame name="frametitolo" src="titolo.htm" noresize>
        <frameset cols="25%,75%">
                <frame name="frameindice" src="indice.htm" noresize>
                <frame name="framecontenuti" src="contenuti.htm" noresize>
        </frameset>

</frameset>

</html>
```

Se, utilizzando i link, si desidera caricare una nuova pagina all'interno di uno dei frame, è sufficiente assegnare all'attributo target del tag <link> il nome del frame in cui la pagina deve comparire.

Esempio:



La pagina è un file di nome Home.htm, così formulata:

```
<html>
<head>
    <title>Esami di Stato 2009/2010</title>
</head>
<frameset rows="20%,80%">
        <frame name="frametitolo" src="titolo.htm" noresize>
        <frameset cols="25%,75%">
                <frame name="frameindice" src="indice.htm" noresize>
                <frame name="framecontenuti" src="DateRiferimenti.htm" noresize>
        </frameset>
</frameset>
</html>
```

Nel frame in alto, di nome ***frametitolo***, compare la pagina Titolo.htm:

```
<html>
<head>
    <title>Titolo</title>
</head>
<body>
        <table align="center" width="90%">
        <tr>
                <td align="left"><img src="./immagini/miur.jpg"></td>
                <td align="center"><font face="Verdana" size="7" color="red">
                Esami di Stato 2009/2010</font></td>
                <td align="right"><img src="./immagini/stud.jpg"></td>
</body>
</html>
```

Nel frame a sinistra, di nome ***frameindice***, compare la pagina Indice.htm

```
<html>
<head>
    <title>Indice</title>
</head>
<body>
<font face="verdana">
<font color="red"><b>Sommario</b></font>
<ul type="circle">
<li><a href="materie.htm" target="framecontenuti">Materie</a>
<li><a href="commissari.htm" target="framecontenuti">Commissari</a>
<li><a href="materiale.htm" target="framecontenuti">Materiale di supporto per le commissioni</a>
<li><a href="requisiti.htm" target="framecontenuti">Requisiti per la partecipazione</a>
<li><a href="esterni.htm" target="framecontenuti">Candidati esterni </a>
<li><a href="ammissione.htm" target="framecontenuti">Criteri di ammissione</a>
<li><a href="provescritte.htm" target="framecontenuti">Le prove scritte</a>
<li><a href="primaprova.htm" target="framecontenuti">La prima prova scritta</a>
<li><a href="secondaprova.htm" target="framecontenuti">La seconda prova scritta</a>
<li><a href="terzaprova.htm" target="framecontenuti">La terza prova scritta</a>
<li><a href="colloquio.htm" target="framecontenuti">Il colloquio</a>
<li><a href="colloquio.htm" target="framecontenuti">Tempi e struttura del colloquio</a>
</ul>
</font>
</body>
</html>
```

Nel frame a destra, di nome ***framecontenuti***, compaiono i contenuti. C'è una pagina iniziale, caricata direttamente nella definizione del frameset, così formulata:

```
<html>
<head>
    <title>Date e riferimenti</title>
</head>
<body>
        <font face="Verdana">
        <br>
        <p align="center">La sessione degli esami di Stato conclusivi dei corsi di studio di istruzione secondaria superiore, per l'anno scolastico 2009/2010, avrà inizio <b>martedì 22 giugno 2009 alle ore 8,30</b> con lo svolgimento della <b>prova di Italiano</b>, <b>mercoledì 23 sempre alle ore 8,30</b> sarà eseguita la <b>seconda prova scritta</b> e <b>venerdì 25</b> la <b>terza prova scritta</b> con orario stabilito dalle commissioni (O.M. 74 del 4-8-2009).</p>
```

```
        <p align="center">Tutte le notizie si trovano nel sito della Pubblica Istruzione nella
sezione
        <br>
        <a target="_blanck"
        href="http://www.pubblica.istruzione.it/argomenti/esamedistato/secondo_ciclo/quadro/
        provv_2010.htm#cm11">Il tuo esame di Stato
        </a>
        <br>
        che raccoglie tra l'altro la normativa
        per l'esame di Stato.</p>
        </font>
</body>
</html>
```

Cliccando su uno dei link nel frame a sinistra, viene caricata la pagina corrispondente nel frame a destra.
Ad esempio, cliccando sul link Materie, la pagina è la seguente:

La pagina caricata nel frame Contenuti si chiama Materie.htm ed è così formulata:

```
<html>
<head>
    <title>Contenuti</title>
</head>
<body>
        <font face="Verdana">
        <br>
        <font color="red">
        <h3 align="Center">
        Materie oggetto della seconda prova scritta e materie affidate ai commissari esterni
        </h3>
```

```
        </font>
        <hr color="red" width="70%" align="center">
        <p align="center">L'elenco delle materie scelte per l'esame di stato nei vari corsi
        sarà reso noto appena verrà pubblicato dal MIUR.</p>
        <p align="center">Nel sito esterno del Ministero sarà disponibile inoltre il<br>
        <a href="http://www.trampi.istruzione.it/matEsami/startMotore.do" target="_blanck">
        Motore di ricerca</a><br> in ambiente
        web per l'interrogazione delle materie oggetto
        della seconda prova scritta e di quelle affidate ai commissari esterni.</p>
        <br>
        <br>
        <p align="center"><a href="DateRiferimenti.htm">Home</a></p>
        </font>
</body>
</html>
```

Si osservi che il link Home viene usato per far riapparire la pagina iniziale nel frame dei contenuti.

I motivi di successo dei frames sono essenzialmente due:

- si può mantenere un layout fisso, senza doverlo riproporre in ogni pagina, come con le tabelle;
- si può mantenere sempre visibili le parti fisse del layout (titoli e strumenti di navigazione).

Tuttavia gli svantaggi che comporta un uso scorretto di un layout a frame sono superiori ai vantaggi che possono derivare dal loro utilizzo.

I motori di ricerca, ad esempio, navigano di link in link attraverso il web, per catturare contenuti da indicizzare.
È frequente allora che una struttura a frame sia inserita all'interno di un motore di ricerca in modo errato: a volte viene catturato solo un menu, altre volte compare soltanto la parte interna con il contenuto del frame e dunque viene perso ogni menu di navigazione.
Inoltre l'indirizzo della pagina non esplicita i contenuti visualizzati nei vari frame, per cui la pagina non rispetta i criteri di accessibilità e usabilità definiti dal W3C.
Problemi sono presentati anche dai browser (alcuni stanno comunque rimediando) per l'aggiunta ai preferiti (viene memorizzato solo l'indirizzo della pagina, non il contenuto di ogni frame) e la stampa.

Frequente e giustificabile è invece l'uso dei frame nelle pagine web di accesso riservato, come quelle per l'utilizzo di posta elettronica, dove le problematiche indicate non hanno particolare peso.

I moduli

I moduli (o form) sono usati solitamente per la creazione di pagine per la raccolta di dati.
Con i moduli è possibile aggiungere caselle di testo, pulsanti di opzione, caselle di controllo e altri oggetti di interfaccia utente. Una volta che gli utenti avranno inserito i dati e fatto clic su un pulsante di conferma, il server, per elaborarli, dovrà utilizzare un programma di gestione appositamente creato e scritto in linguaggio opportuno (asp, php, cgi).

Esempio:
Nella home page di Google sono presenti

- *una casella di testo (dove inserire le parole da cercare),*
- *due pulsanti di comando ("Cerca con Google" e "Mi sento fortunato")*

- *tre pulsanti d'opzione ("il web", "pagine in Italiano" e "pagine provenienti da: Italia").*

In HTML l'inizio di un modulo è individuato dal tag **<form>** e la fine da **</form>.** È possibile utilizzare tre attributi:

- **name** - rappresenta il nome del modulo;
- **method** - può assumere due valori, *get* o *post*, che forniscono al server informazioni su come devono essere trasmessi i dati.
- **action** - è un URL che rappresenta l'indirizzo di uno script per l'elaborazione dei dati trasmessi al server tramite il modulo.
- **enctype** - specifica come il browser codifica i dati prima di inviarli al server; tale attributo può essere esplicitato solo se si usa il metodo POST. L'attributo enctype può assumere due valori:
  - *application/x-www-form-urlencoded* (è il valore di default). Lo spazio è convertito in "+" e gli altri caratteri speciali (%, £ , $, @, ?, ecc..) sono codificati utilizzando il codice ascii (espresso in Esadecimale).
    Ad esempio, la stringa matteo@google.com viene codificata in matteo%40google.com. Questo perchè il carattere speciale @ si rappresenta con il valore decimale 64 del codice Ascii, cioè valore 40 nel sistema esadecimale. Prima di codificare i caratteri speciali, il browser inserisce il simbolo %.
  - *multipart/form-data*, permette di inviare i dati del form come una sequenza di varie parti. Ogni parte può contenere un tipo di contenuto diverso. Questo valore deve essere usato nel caso in cui l'utente vuole fare l'upload di file sul server.

Esempio:

<form name="frmOrdineAcquisto" method="get" action="order.asp" enctype="application/x-www-form-urlencoded">

….

</form>

order.asp è, in questo esempio, un file che contiene codice (script) in linguaggio ASP per la gestione dei dati trasmessi.

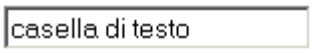
In un modulo (e quindi tra i tag <form> e </form>), per acquisire dati dall'utente, possono essere inseriti i classici oggetti per le interfacce grafiche.

Casella di testo

In HTML una casella di testo è individuata dal tag **<input>** con l'attributo **type** uguale a **text**. Oltre all'attributo **type**, che viene usato per definire l'oggetto che viene posto all'interno del form (casella di testo, pulsante di opzione, etc.), si possono usare i seguenti attributi:

- **name** - fornisce alla casella di testo un nome per identificarla.
- **maxlength** - imposta la lunghezza massima in caratteri consentita dal campo.
- **size** - controlla l'ampiezza in caratteri della casella sulla pagina.
- **value** - è il valore restituito; può essere usato per inizializzare.

Esempio: <input type="text" name="txtcasella" maxlength="20" size="20" value="casella di testo">

fa comparire la casella 

Casella password

È un caso particolare di casella di testo. Ponendo **type**="**password"**, la casella di testo utilizza asterischi per nascondere l'input. Tutti gli altri attributi restano invariati.

Campo nascosto

Potremmo avere la necessità di passare parametri "di servizio", senza far percepire la loro presenza all'utente. In questo caso possiamo utilizzare campi nascosti, presenti all'interno del form ma invisibili all'utente (è importante specificare la coppia "nome-valore"). In questo caso si attribuisce il valore hidden all'attributo type del tag <input>.

Esempio:

<input type="hidden" name="urlDiProvenienza" value="www.html.it">

Pulsante di comando

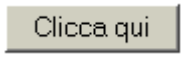
I pulsanti utilizzano anche il tag **<input>** e, a seconda del valore assegnato all'attributo **type**, possono essere di tre tipi:

- **type="submit"** dà luogo ad un pulsante cliccando sul quale si inviano i dati del modulo allo script;
- **type="reset"** dà luogo ad un pulsante che ripristina il valore di default degli oggetti del modulo;
- **type="button"** è un pulsante generico, il cui comportamento non è predefinito, come negli altri due casi, ma deve essere programmato (in linguaggi come Java Script e VB Script).

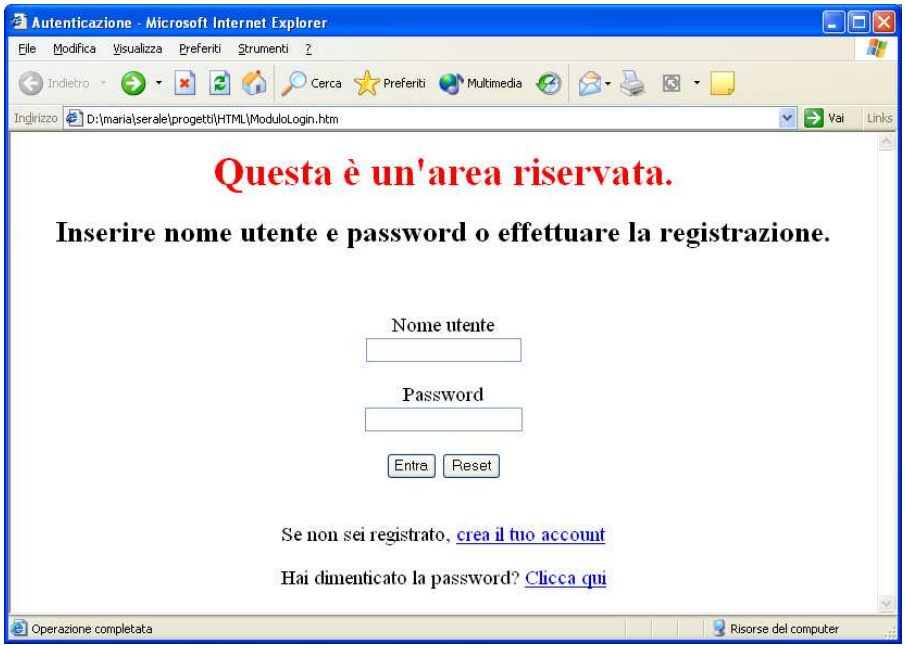
Tutti e tre usano l'attributo **Name** per identificare il pulsante nel codice e l'attributo **Value** per specificare il testo che compare sul pulsante.

Esempio:

<input type="submit" name="btnPulsante1" value="Clicca qui">

Fa comparire il pulsante 

Esempio:




```
<html>
<head>
<title>Autenticazione</title>
</head>
<body>
        <h1 align="center"><font color="red"> Questa è un'area riservata.</font></h1>
        <h2 align="center">Inserire nome utente e password o effettuare la registrazione.</h2>
        <br><br>
        <div align="center">
            <form name="frmModulo" method="post" action="login.asp">
                Nome utente <br><input type="text" name="txtNomeUtente" maxlength="20"
                  size="20"><br>
                Password<br><input type="password" name="CasellaPassword" maxlength="20"
                  size="20"><br><br>
                <input type="submit" name="PulsanteEntra" value="Entra">
                <input type="reset" name="PulsanteReset" value="Reset"><br><br>
              </form><br><br>
            Se non sei registrato, <a href="registrati.htm">crea il tuo account</a><br><br>
            Hai dimenticato la password? <a href="ricordapassword.htm">Clicca qui</a>
        </div>
</body>
</html>
```

Casella di testo multiriga (Text Area)

Il tag HTML che individua una casella di testo a più righe è **<textarea>**, che richiede il tag di chiusura **</textarea>**.
Tutto quello che compare fra il tag di apertura e quello di chiusura è il testo di default che appare nella casella di testo.
Gli attributi sono:

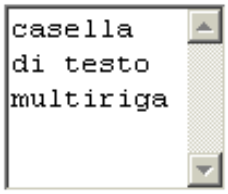
- **name** - fornisce alla casella di testo un nome per identificarla.
- **rows e cols** - permettono rispettivamente di impostare il numero di righe e di colonne.
- **wrap** - specifica il comportamento della textarea nel caso in cui l'utente scrive un contenuto avente una larghezza maggiore di quella della textarea. L'attributo wrap può assumere tre valori:

- *off*, che indica che verrà aggiunta la scrollbar orizzontale, se l'utente inserisce del testo più largo della larghezza della TextArea;
- *virtual o soft*, che è il valore default. Indica che il testo andrà a capo automaticamente, se l'utente inserisce del testo più largo della larghezza della TextArea; però il server riceverà una sola linea di testo.
- *physical o hard*, indica che il tasto andrà a capo automaticamente, se l'utente inserisce del testo più largo della larghezza della textarea; però il server riceverà tante linee di testo, cosi come appaiono nella TextArea.

Esempio:

```
<form>
    <textarea name="testolungo" rows="14" cols="10" wrap="physical" >
        casella di testo  multiriga
    </textarea>
</form>
```

fa comparire la casella multiriga:



Casella di controllo (Check Box)

Se nel tag **<input>** l'attributo **type** è **checkbox**, nel form compare una casella di controllo:

Gli attributi sono:

- **name** - fornisce alla casella di controllo un nome per identificarla.
- **value** - è il valore che viene restituito quando la casella è spuntata; restituito è **on**; se la casella non è spuntata, nessun valor
- **checked** – fa apparire la casella selezionata (con il segno di sp
- **unchecked** - fa apparire la casella deselezionata (senza il segn

Name="Controllo[]" se c'è un gruppo di checkbox che vogliamo usare come vettore

Esempio:

```
<form>
    Certifico di essere maggiorenne e di aver letto e accettato le condizioni d'uso del servizio
    <input type="checkbox" name="controllo" value="yes" checked>
</form>
```

fa comparire:

Certifico di essere maggiorenne e di aver letto e accettato le condizioni d'uso del servizio 

Pulsante di opzione (Radio Button)

Se nel tag **<input>** l'attributo **type** vale **radio**, nel form compare un pulsante di opzione:

Gli attributi sono:

- **name** - fornisce al pulsante di opzione un nome per identificarlo; tutti i pulsanti di opzione che fanno parte dello stesso gruppo devono avere lo stesso nome, altrimenti si potrebbero avere più pulsanti selezionati nello stesso gruppo (invece uno solo deve essere selezionato).

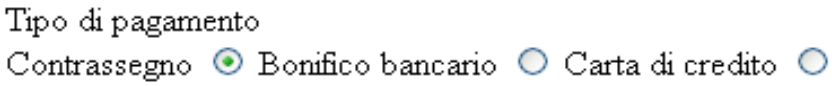
- **value** - è il valore che viene restituito quando il pulsante è selezionato; se omesso, viene restituito il valore **on**; se il pulsante non è selezionato, nessun valore è inviato.

- **checked** - seleziona il pulsante (fa apparire il puntino all'interno).

Esempio:

```
<form>
   Tipo di pagamento<br>
   Contrassegno <input type="radio" name="TipoPagamento" value="Contrassegno" checked>
   Bonifico bancario <input type="radio" name="TipoPagamento" value="Bonifico bancario" >
   Carta di credito <input type="radio" name="TipoPagamento" value="Carta di credito" >
</form>
```

fa comparire



Elenchi a discesa (List e Combo Box)

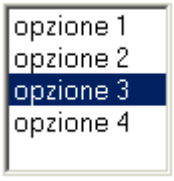
In HTML il tag che individua un elenco a discesa è **<select>**, che richiede il tag di chiusura **</select>** ed usa la seguente sintassi:

```
<select name="nomeselezione" [size=elementi visibili] [multiple]>
      <option value="valore_1" [selected]>opzione 1 </option>
      ...
      <option value="valore_n" [selected]>opzione n </option>
</select>
```

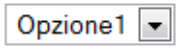
I tag **option** introducono le opzioni della lista.

Gli attributi **size**, **multiple** e **selected** sono facoltativi (per questo sono indicati tra parentesi quadre).

L'attributo **size** determina il numero di opzioni visibili nella lista; questo numero determina la lunghezza della lista e, quando il numero delle opzioni supera la lunghezza della lista, esse non sono più visibili se non usando la barra di scorrimento, che appare automaticamente. Nel seguente esempio, size vale 5, per cui quattro opzioni sono perfettamente visibili.



Con size=0 (default), appare una casella combinata, del tipo:



L'attributo **multiple** controlla se si possono selezionare più elementi contemporaneamente.
Per ogni opzione selezionata, il valore inviato è specificato tramite l'attributo **value**.
L'attributo **selected** indica una voce di menu selezionata.

Esempio:

```
<form>
      <select name="Puglia" size="0">
            <option value="Bari" selected>Bari</option>
```

```
            <option value="Brindisi">Brindisi</option>
            <option value="Foggia">Foggia</option>
            <option value="Lecce">Lecce</option>
            <option value="Taranto">Taranto</option>
        </select>
    </form>
```

fa comparire la casella combinata:

L'elenco appare cliccando sulla freccia a destra.

Con size="5", appare invece la lista completa, con la voce Bari selezionata, come indicato nel codice:

File upload control

Il tag **<input type="file">** permette di eseguire la selezione di un file di cui si vorrà fare l'upload sul server.
L'attributo **accept**, indica il tipo di file di cui si vuole fare l'upload. Esempi di valori di tale attributo possono essere i seguenti:

- **estensione_del_file**
- **audio/***
- **video/***
- **image/***

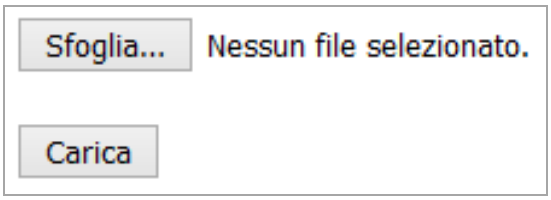
L'attributo **size** indica invece la larghezza del campo.

Nel modulo appare il pulsante "sfoglia" o "browse" (a seconda della lingua del browser dell'utente).

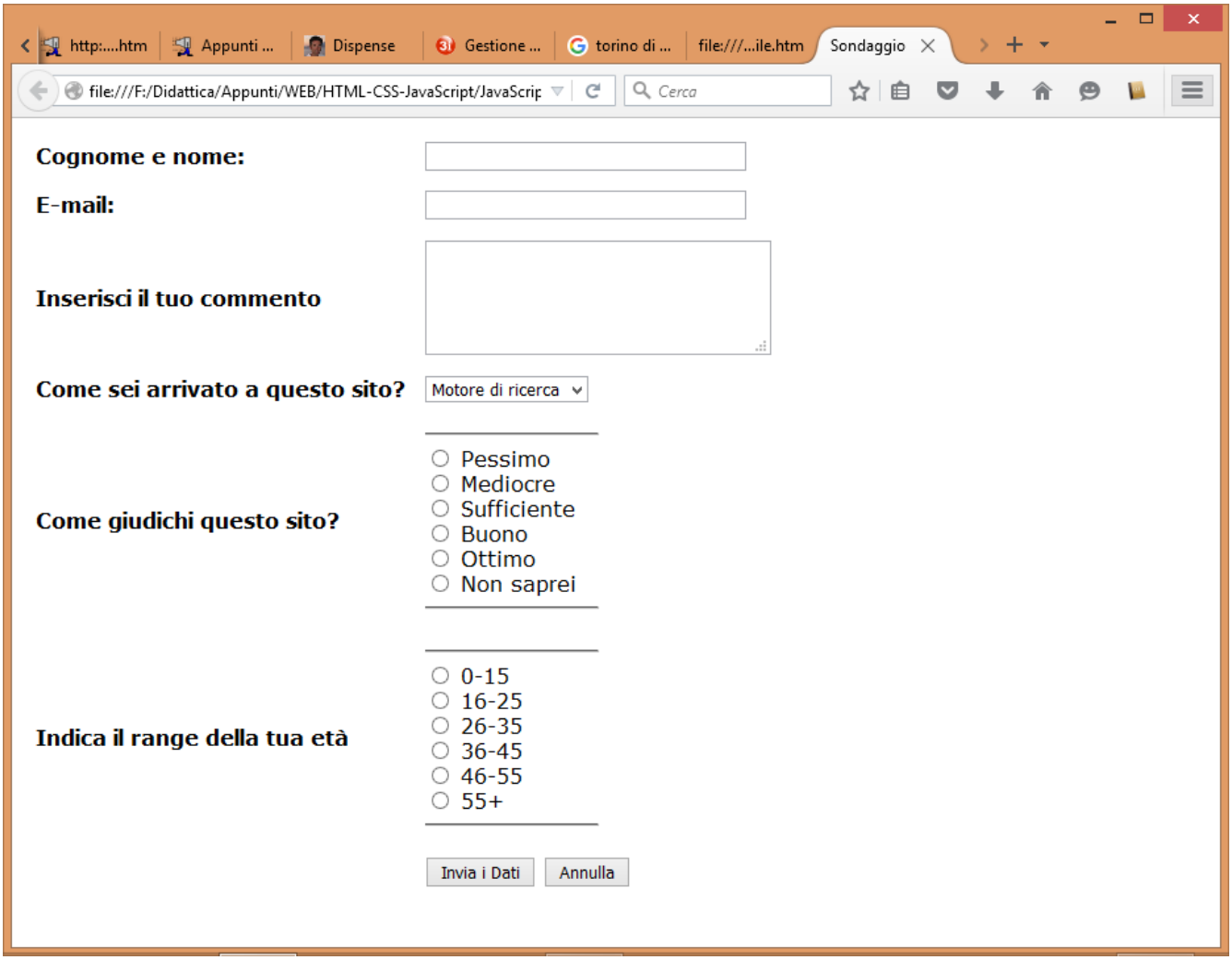
Esempio:

```
<form action="carica.asp" method="post" enctype="multipart/form-data" name="frmUpload">
    <input type="file" name="my_file" id="my_file" /><br><br>
    <input type="submit" name="btnCarica" value="Carica" />
</form>
```

che dà



Esercizio: *Realizzare la seguente pagina, contenente un modulo.*




```
<html>

<head>
   <title>Sondaggio</title>
</head>

<body text="black">

<form name="FrmSondaggio" action="RilevaDati.asp" method="post">
        <font face="verdana">

        <table border="0" cellspacing="5" cellpadding="5">
                <tr>
                        <td><b>Cognome e nome:</b></td>
                        <td>
                                <input type="text" name="TxtNome" size="35" maxlength="40" value="">
                        </td>
                </tr>
                <tr>
```

```
        <td><b>E-mail:</b></td>
        <td>
                <input type="text" name="TxtEmail" size="35" maxlength="40" value="">
        </td>
</tr>
<tr>
        <td valign="middle"><b>Inserisci il tuo commento</b></td>
        <td>
                <textarea name="TxtCommento" rows="4" cols="30"></textarea>
        </td>
</tr>
<tr>
        <td><b>Come sei arrivato a questo sito?</b></td>
        <td>
                <SELECT name="SelComeArrivato">
                  <OPTION SELECTED>Motore di ricerca</OPTION>
                  <OPTION>Banner</OPTION>
                  <OPTION>Link su altro sito</OPTION>
                  <OPTION>Guestbook</OPTION>
                  <OPTION>News Group</OPTION>
                  <OPTION>Casualmente</OPTION>
                  <OPTION>Altro</OPTION>
                </SELECT>
        </td>
</tr>
<tr>
        <td><b>Come giudichi questo sito?</b></td>
        <td>
                 <hr width="50%" align="left">
                 <input type="radio" name="RdGiudizio" value="pessimo"> Pessimo<br>
                 <input type="radio" name="RdGiudizio" value="mediocre"> Mediocre<br>
                 <input type="radio" name="RdGiudizio" value="sufficiente"> Sufficiente<br>
                 <input type="radio" name="RdGiudizio" value="buono"> Buono<br>
                 <input type="radio" name="RdGiudizio" value="ottimo"> Ottimo<br>
                 <input type="radio" name="RdGiudizio" value="non saprei"> Non saprei<br>
                 <hr width="50%" align="left">
        </td>
</tr>
<tr>
  <td><b>Indica il range della tua et&agrave;</b></td>
  <td>
                <hr width="50%" align="left">
                <input type="radio" name="RdEta" value="0-15"> 0-15<br>
                <input type="radio" name="RdEta" value="16-25"> 16-25<br>
                <input type="radio" name="RdEta" value="26-35"> 26-35<br>
                <input type="radio" name="RdEta" value="36-45"> 36-45<br>
                <input type="radio" name="RdEta" value="46-55"> 46-55<br>
                <input type="radio" name="RdEta" value="55+"> 55+<br>
                <hr width="50%" align="left">
        </td>
</tr>
<tr>
        <td></td>
```

```
                    <td>
                        <input type="submit" value="Invia i Dati" name="CmdInvia">
                        <input type="reset" value="Annulla" name="CmdReset">
                    </td>
                </tr>
        </table>

        </font>
 </form>

 </body>

</html>
```

I tag <fieldset> e <legend>

Per la loro natura di "raccoglitori di informazioni", i moduli tendono a ingigantirsi e diventare lunghissimi. Per questo sono stati introdotti alcuni tag per fare un po' d'ordine all'interno dei form.
Il tag <fieldset> viene utilizzato all'interno di un form per raggruppare vari oggetti tra loro, tracciando un box intorno a un gruppo di elementi e dividendo così il form in macro-aree. Il tag <legend> consente invece di indicare il nome di ciascuna macro-area.

La sintassi è:

```
<form>
<fieldset>
  <legend>Titolo1</legend>
   …
</fieldset>

<fieldset>
  <legend>Titolo2</legend>
  …
</fieldset>

…

</form>
```

Esempio:

```
<form action="test.php" method="post">
  <fieldset><legend>
        Informazioni sull'utente:</legend>
        Cognome<br>
        <input type="text" value="Cognome" size="25"><br>
        Nome<br>
        <input type="text" value="Nome" size="25"><br>
        Et&agrave;<br>
        <input type="text" value="Eta" size="6"><br>
        Data di nascita<br>
        <input name="text" type="text" value="DataN" size="8">
  </fieldset>
</form>
```

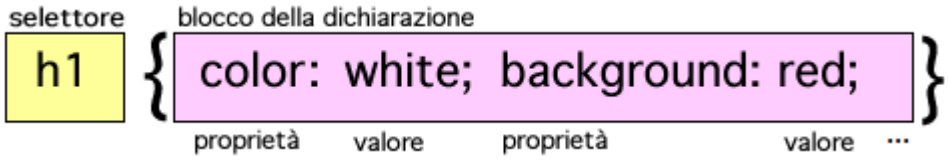
I fogli di stile (CSS)

I **Cascading Style Sheets** (**CSS**), in Italiano **fogli di stile**, sono un potente strumento per formulare e gestire il layout di un sito web. Essi sono inclusi in una o più pagine HTML per controllare in maniera automatica e veloce l'aspetto e lo stile della pagina. Inoltre consentono di ottenere effetti che non sarebbero possibili con l'uso del solo linguaggio HTML. Il termine "cascata" (cascading) sta a indicare una delle caratteristiche principali dei CSS, il fatto di poter essere inseriti più volte in uno stesso documento seguendo regole gerarchiche precise, quindi ci sono CSS più importanti e CSS meno importanti.
Si farà riferimento alla versione CSS3.

I CSS sono uno standard riconosciuto dal W3C e un sito di qualità non può non farne uso.

Compito dei fogli di stile è ridefinire e/o potenziare i tag HTML, attraverso un insieme di regole che offrono anche il grande vantaggio di modificare, in un colpo solo, le impostazioni in una o più pagine.

Vediamo com'è fatta una regola:



Essa è composta da due blocchi principali:
- **il selettore**
- **il blocco delle dichiarazioni**

Il selettore serve a definire la parte del documento cui sarà applicata la regola. In questo caso, ad esempio, saranno formattati tutti gli elementi **<h1>**: lo sfondo sarà rosso, il colore del testo bianco.
Il blocco delle dichiarazioni è delimitato da **due parentesi graffe**. Al suo interno possono trovare posto più dichiarazioni. Esse sono sempre composte di una coppia:
- **proprietà**
- **valore**

La proprietà definisce un aspetto dell'elemento da modificare (colore di sfondo, bordo, dimensione, etc) secondo il valore espresso. Proprietà e valore devono essere separati dai **due punti**.
Se in un blocco si definiscono più dichiarazioni, come nell'esempio in figura, esse vanno separate dal **punto e virgola**. Il linguaggio non impone che si metta il punto e virgola dopo l'ultima dichiarazione, ma alcuni browser più datati lo richiedono: conviene aggiungerlo sempre per sicurezza e per una maggiore compatibilità.

Gli spazi bianchi lasciati all'interno di una regola non influiscono sul risultato. Il consiglio è di lasciare sempre uno spazio tra le varie parti, per una migliore leggibilità.

Si possono inserire parti di commento in un CSS racchiudendole tra i segni:
/* come segno di apertura
***/** come segno di chiusura

Applicazione delle regole di stile

Le regole di stile possono essere applicate a un documento html in tre modi:

- **In linea**

Le informazioni sullo stile sono specificate per ogni singolo elemento all'interno del tag HTML, utilizzando l'**attributo style**; in questo modo è possibile definire o ridefinire alcune proprietà del tag stesso.
All'**attributo style** è assegnata una stringa in cui sono elencate le regole.

Esempio: *<h1 style="color:blue; font-size:20pt;">L'intestazione è di colore blu</h1>*

- **Integrato (o interno)**

Il foglio di stile è inserito all'inizio del documento HTML, utilizzando il **tag style** all'interno della sezione head del documento.
Attributo del **tag style** è **type**, cui è assegnato il valore "text/css".

Esempio: *Il codice seguente ridefinisce gli stili dei tag h1 e p; queste modifiche saranno valide per tutte le occorrenze dei tag h1 e p nella pagina.*

```
<html>
  <head>
      <title>…</title>
      <style type="text/css">
          h1 {color: blue; font-size:20pt;}
          h2 {color:grey; font-size:10px;}
      </style>
  </head>
  <body>

  </body>
</html>
```

- **Esterno**

Le regole di stile sono memorizzate in un documento separato, che è un **file con estensione .css** ed è richiamato dal documento html in due modi:

- un link ipertestuale al foglio di stile, usando il tag link nella sezione head della pagina

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

- attraverso l'uso del comando **@import** all'interno del tag **style**, sempre nella sezione head della pagina:

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

È consigliabile utilizzarli entrambi, perché non tutti i browser riconoscono l'uno e l'altro.

Che tipo di foglio di stile scegliere? Dipende dal tipo di sito, dal numero di pagine da cui è formato, se le pagine devono avere tutte o gran parte di esse lo stesso stile. Si deve in tal senso considerare che:

- i fogli di stile "in linea" non sono molto pratici, in quanto per ogni tag interessato occorre inserirli manualmente, operazione che su pagine complesse può diventare molto lunga e ripetitiva;
- i fogli di stile "interni" sono adatti a siti formati da poche pagine oppure a siti che, pur avendo un certo numero di pagine, necessitano di stili differenti per ciascuna pagina o gruppo di pagine;

- i fogli di stile "esterni" sono adatti a siti di dimensioni medio grandi o grandi e che hanno l'esigenza che tutte le pagine abbiano, in termini di stile, le medesime caratteristiche. Con un foglio di stile esterno è, infatti, molto facile cambiare le caratteristiche di tutte le pagine.

Selettori

Fondamentalmente una regola CSS è applicata a un **selettore**. I tipi più importanti di selettori sono:

- ***Selettore a livello di elemento o selettore di tipo***

  Il modo più semplice per un selettore di riferirsi all'elemento html è tramite il tag:

  Esempio:
  h1 {color: blue}

  Questo tipo di selettori può essere raggruppato in una lista separata da virgole, cosicché una singola proprietà può essere applicata a tutti gli elementi della lista.

  Esempio:
  h1,h2,p {color: blue}

- **Selettore universale *(universal selector)***
  Usando come selettore il simbolo *, detto selettore universale, le regole specificate saranno applicate a tutti gli elementi che ne prevedano un'efficacia.
  Il selettore universale, sebbene molto potente, non dovrebbe essere utilizzato se non in casi molto specifici, in quanto ci sono modi più convenienti per ottenere lo stesso risultato.

  Esempio:
  ** {color: green;}*
  attribuisce colore del testo verde a tutti gli elementi di una pagina html

- **Selettore contestuale o di discendenti**

  Specifica lo stile in base al contesto in cui il selettore viene posto.

  Esempio:
  li b {color: blue;}
  Il testo in grassetto degli elementi della lista è rappresentato in blue.

- **Selettori di classe**

  Per uno stesso tag è possibile definire una o più classi, a ciascuna delle quali corrisponde una diversa definizione delle proprietà.

  Esempio:
  h1.nomeclasse1{color: blue;}
  h1.nomeclasse2{color: red;}

  Usando l'attributo **class** del tag, cui si assegna il nome di una classe, è poi possibile richiamare lo stile definito per quella classe:

  Esempio:
  <h1 class="nomeclasse1">Titolo</h1>

È anche possibile definire una classe senza attribuirla a un tag in particolare; basta omettere il nome del tag nella definizione della classe o far precedere il selettore universale * al punto.

Esempio:
.nomeclasse{color:blue;} *oppure* **.nomeclasse{color:blue;}*

In tal modo la classe potrà essere utilizzata da un qualsiasi tag.

- **Selettore di ID *(id selector)***

I selettori di ID permettono di assegnare una regola css a un **unico elemento** nella pagina html che è specificato attraverso l'attributo id.
La relativa regola css va specificata anteponendo al nome dell'id il simbolo **#**.

Esempio:

Con la regola
#titolo {color: blue;}
si assegna il colore blue all'elemento che presenti questa definizione nel codice HTML, ad esempio:
<h1 id="titolo">...</h1>

Come per le classi è possibile usare una sintassi con elemento:
p#nome_id
In realtà questa modalità è superflua: con un id univoco, non abbiamo alcun bisogno di distinguere l'elemento cui verrà applicata.

- **Selettore contestuale o di discendenti**

Specifica lo stile in base al contesto in cui il selettore viene posto.

Esempio:
li b {color: blue;}
Il testo in grassetto degli elementi della lista è rappresentato in blue.

- **Selettore di figli**

Il selettore di figli (**>**) consente di selezionare un elemento che è figlio diretto dell'elemento padre.
La Sintassi è

```
padre > figlio {dichiarazioni;}
```

Questo selettore è solo in apparenza simile al selettore di discendenti. La differenza sta nella relazione di discendenza tra gli elementi, che in questo caso deve essere di **primo livello**, cioè non deve avere tag contenitori intermedi.

Esempio:

```
<div id="box">
 <p>Primo paragrafo</p>
 <div>
  <p>Secondo paragrafo</p>
 </div>
 <p>Terzo paragrafo</p>
</div>
```

Dei tre paragrafi solo il primo e il terzo sono figli diretti del div con id #box. Il secondo è invece figlio diretto di un elemento div anonimo. Tutti e tre, però, sono discendenti del div con id #box.

Pertanto, con la seguente dichiarazione, solo il primo e il terzo paragrafo avranno il testo bianco:

#box > p {color: white}

È opportuno spiegare due fondamentali regole di base.

1. I valori di una proprietà non vanno **mai messi tra virgolette**.
   Uniche eccezioni sono:
   - i valori espressi da stringhe di testo;
   - i nomi dei font formati da più di una parola (esempio: "Times New Roman").
2. Quando si usano valori numerici con unità di misura, non bisogna lasciare spazio tra numero e sigla dell'unità. È quindi
   - corretto scrivere **15px**.
   - sbagliato scrivere **15 px**.

   Lo stesso vale per i valori percentuali: **60%** va bene, **60 %** no. In questi casi la regola sarà ignorata o mal interpretata.

3. Le unità di misura usate per definire dimensioni, spazi o distanze sono essenzialmente tre:
   - **pt (points**, cioè **punti)**: unità di misura tipografica destinata essenzialmente a definire la dimensione dei font.
   - **em (em-height**, cioè **altezza media)**: unità di misura spesso usata dagli autori CSS. 1em equivale all'altezza media di un carattere per un dato font.
   - **px (pixels)**: unità di misura ideale su monitor. E' quella più usata e facile da comprendere.

L'**unità pt è assoluta**, come anche cm, mm, inch; le unità assolute si usano quando il supporto di visualizzazione ha dimensioni fisse, come un foglio A4; sono, infatti, usate per definire fogli di stile quando si deve produrre una versione stampabile della pagina. **Le unità em e px sono relative**, px alla dimensione e alla risoluzione dello schermi, em alla dimensione dei caratteri dell'impostazione attuale del font in uso.
I valori URL, cioè gli indirizzi che puntano a documenti esterni (in genere immagini, come negli sfondi) devono essere introdotti dalla parola chiave **url** e scritti tra parentesi tonde, senza virgolette.

Esempio : url(./immagini/sfondo.gif)

Principali proprietà

Colori e sfondi

Le proprietà dei CSS permettono di specificare il colore del testo e lo sfondo di un elemento. Gli sfondi possono essere colori o immagini.

- La proprietà **color**

Descrive il colore del testo di un elemento. Il valore può essere sia una parola chiave che una specifica *RGB* di colore.

Esempio:

p {color: #ff0000}
definisce il colore rosso per il testo del paragrafo

- Le proprietà che riguardano lo sfondo sono: **background-color, background-image, background-repeat, background-attachment, background-position e background**.

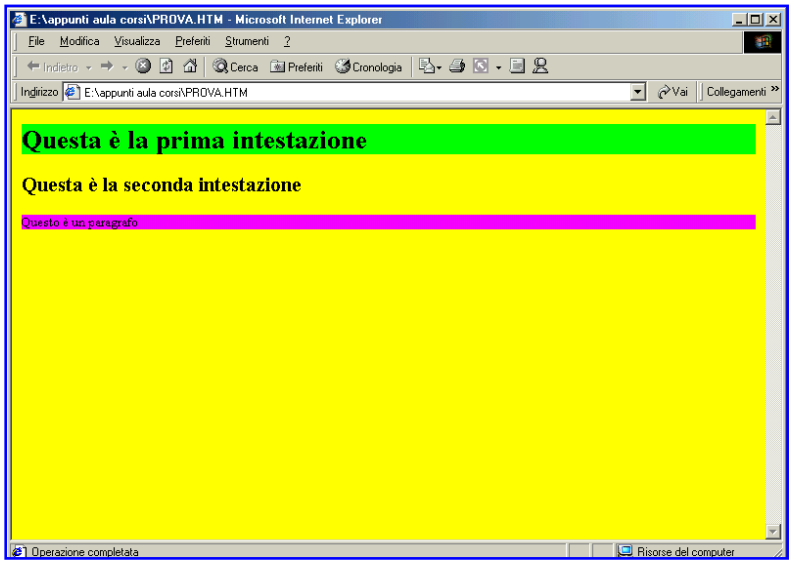
background-color imposta il colore di sfondo di un elemento. Il valore **transparent** non copre lo sfondo.

Esempio:

p {background-color: #0f0f0f; color: #AA0000;}
definisce, per il testo del paragrafo, il colore dello sfondo del testo (una tonalità di grigio) e il colore dei caratteri (una tonalità di rosso).

Esempio: Impostiamo il colore dello sfondo dei titoli e del paragrafo.

```
<html>
 <head>
  <style type="text/css">
        body {background-color: yellow}
        h1 {background-color: #00ff00}
        h2 {background-color: transparent}
        p {background-color: rgb(250,0,255)}
  </style>
 </head>
 <body>
  <h1>Questa &egrave; la prima intestazione</h1>
  <h2>Questa &egrave; la seconda intestazione</h2>
  <p>Questo &egrave; un paragrafo</p>
 </body>
</html>
```



background-image: imposta un'immagine di sfondo per un elemento. Quando si usa un'immagine di sfondo, si dovrebbe anche specificare un colore di sfondo, nel caso in cui l'immagine non fosse reperibile. I valori possibili sono o un URL, per specificare l'immagine, o **none**, per non specificarne alcuna:

Esempio:

H1 {background-color: lightred; background-image: url(sfondo.gif); color: black;}

background-repeat: specifica, nel caso sia stata impostata un'immagine di sfondo, se l'immagine deve essere ripetuta e come. I valori sono:

repeat
L'immagine è ripetuta orizzontalmente e verticalmente.

repeat-x
L'immagine è ripetuta solo orizzontalmente.

repeat-y
L'immagine è ripetuta solo verticalmente.
no-repeat
L'immagine non è ripetuta.

Esempio:

```
body {background-color: white; background-image: url(pattern.gif);
                                        background-repeat: repeat-x; color: blue;}
```

background-attachment: se viene specificata un'immagine di sfondo, questa proprietà specifica se resta fissa rispetto al viewport (valore **fixed**) o se scrolla insieme con il blocco contenitore (**scroll**). Anche se l'immagine è fissa, è visibile solo quando si trova nelle aree di contenuto, padding o bordo dell'elemento.
Esempio:

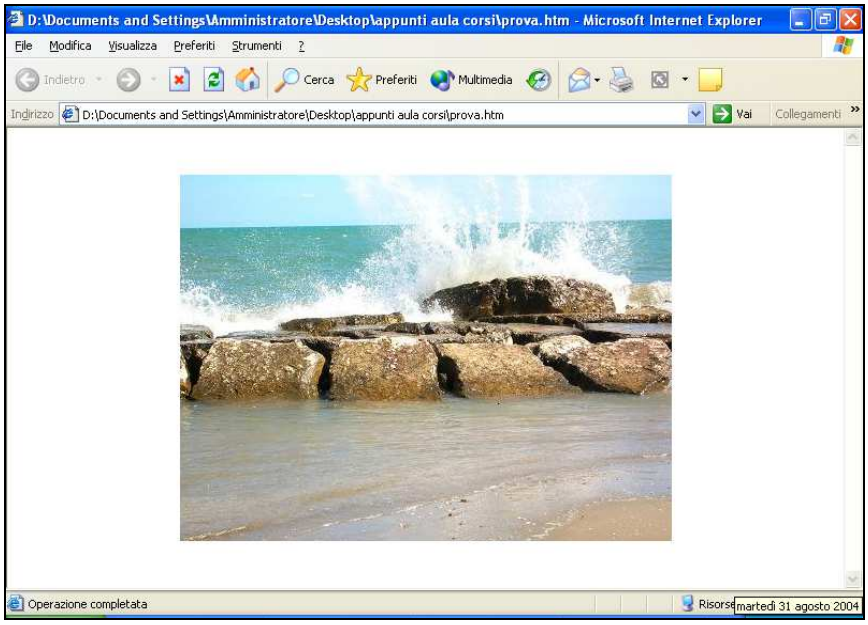
```
body {background-color: #AAAAAA; background-image: url(fiore.jpg);
    background-repeat: no-repeat; background-attachment: fixed; color: yellow;}
```

I browser che non supportano 'fixed' ignorano la parola chiave.

background-position: se viene specificata un'immagine di sfondo, questa proprietà indica la posizione in cui deve apparire. I valori possibili sono diversi, tuttiperò definiscono le coordinate di un punto rispetto al margine sinistro e al margine in alto, il che può avvenire nei seguenti modi:

- con valori in **percentuale**
  Esempio: background-position: 10% 50%.
- con valori espressi in base a specifiche **unità di misura**, che sono **pixel** (**px**), **centimetri** (**cm**), **millimetri** (**mm**), **punti** (**pt**), **pollici** (**in**)
  Esempio: background-position: 50px 50px.
- con le parole chiave **top**, **left**, **bottom**, **right**, **center**.

Esempio: Impostare un'immagine di sfondo al centro della pagina




```
<html>
    <head>
      <style type="text/css">
        body {background-image: url("./mare.jpg"); background-repeat: no-repeat;
            background-attachment: fixed; background-position: center center;}
      </style>
    </head>
  <body>
```

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

background è una proprietà abbreviata che serve a impostare le proprietà dello sfondo in un'unica dichiarazione. Imposta prima le proprietà individuali sul loro valore iniziale e poi assegna valori espliciti dati nella dichiarazione.

Esempio:

```
body {background: #f00aff; color: #000000;}
p {background: #ffcccc url("img.gif") no-repeat fixed 5% 5%;}
```

Font

- La proprietà **font-family**

Il valore è un elenco, in ordine di priorità, di nomi di font. A differenza di altre proprietà CSS, i valori sono separati da virgole per indicare che sono alternativi.

Esempio:

```
body {font-family: Helvetica, Arial, sans-serif}
```

I nomi di font che contengono uno spazio devono essere racchiusi tra virgolette.

Esempio:

```
body {font-family: "Book Antiqua", Palatino, Georgia, serif}
```

- La proprietà **font-style**

La proprietà 'font-style' determina se una famiglia di font deve essere visualizzata con caratteri corsivi o obliqui. I valori possibili sono **normal** (normale) o **italic** (corsivo).

Esempi:

```
h1 {font-style: normal}
h1 {font-style: italic}
```

- La proprietà **font-variant**

Un altro tipo di variante all'interno di una famiglia di font è il maiuscoletto (small-caps). In un font in maiuscoletto le lettere minuscole sono simili a quelle maiuscole, ma con una dimensione minore e con proporzioni leggermente differenti. Il valore **small-caps** crea questa variante:

Esempio:

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

- La proprietà **font-weight**

La proprietà font-weight seleziona il peso di un font. I valori vanno da **100** a **900** e formano una sequenza ordinata, in cui ogni numero indica un peso che è scuro almeno come il suo predecessore. La parola chiave **normal** corrisponde a 400 e **bold** (grassetto) corrisponde a 700. Nella quasi totalità dei casi, tuttavia, i browser non riescono a coprire i valori oltre 700, poiché i font che dispongono di caratteri con un peso superiore a Bold non sono molto diffusi (per esempio Heavy, Black o Extrablack).

Esempi:

```
strong {font-weight: 400} /* normal */
h4 {font-weight: bold}
h5 {font-weight: 600}
```

- La proprietà **font-size**

Serve a impostare la dimensione del testo.

Esempio:

p {font-size:12px;}

I valori della dimensione del testo possono essere espressi in modo assoluto o relativo.

Sono **valori assoluti**:

- le sette parole chiave
  xx-small, **x-small**, **small**, **medium**, **large**, **x-large**, **xx-large**.
  La seguente tabella mostra una corrispondenza tra le parole chiave, le intestazioni HTML e i valori dell'attributo size del tag <font>. Il valore 'medium' è usato come valore intermedio di riferimento.

Parola chiave	Intestazione HTML	Valore di 'size'
xx-small	h6	1
x-small	-	-
small	h5	2
medium	h4	3
large	h3	4
x-large	h2	5
xx-large	h1	6
-	-	7

- I valori espressi con le unità di misura, delle quali quella più idonea per il testo è px. In termini di em, invece, il riferimento è **medium = 1em**

Sono **valori relativi**:

- le parole chiave **smaller** e **larger**;
- valori espressi in **percentuale**.

I valori relativi smaller e larger fanno riferimento alla dimensione attiva fino all'impostazione. Ad esempio, se la dimensione corrente è pari a medium, il valore larger renderà la dimensione del font dell'elemento corrente come large, e smaller come small.
Per i valori percentuali, il riferimento è **medium=100%=1em**.

Esempi:

h2 {font-size:15px}
p {font-size:100%}

- La proprietà **font**

La proprietà font è una proprietà abbreviata che imposta le proprietà dei font in un'unica dichiarazione.

Esempio:

h1 {font: italic small-caps bold 2em/1 Georgia, Arial}

I valori non dichiarati esplicitamente assumono il loro valore iniziale.

Testi

- La proprietà **text-align**

Imposta l'allineamento del testo. I valori possibili sono:
- **left** - allinea il testo a sinistra
- **right** - allinea il testo a destra
- **center** - centra il testo
- **justify** - giustifica il testo

- La proprietà **text-decoration**

Imposta particolari decorazioni e stili per il testo. I valori possibili sono:
- **underline** - Il testo sarà sottolineato.
- **overline** - Il testo avrà una linea superiore.
- **line-through** - Il testo sarà attraversato da una linea orizzontale al centro.
- **none** - Non produce alcuna decorazione.
- **Blink** - Il testo lampeggia. Scarsissimo il supporto dei browser, perché il testo lampeggiante è sconsigliato nelle pratiche di accessibilità dei contenuti web.

- La proprietà **text-indent**

Imposta l'indentazione della prima riga di un elemento. Si accettano valori negativi.

Esempio:

```
p {text-indent:5px; text-align:justify; background-color:yellow; }
```

- La proprietà **letter-spacing**

Imposta la spaziatura tra i caratteri. Sono ammessi valori negativi. Con il valore **normal** la spaziatura è quella normale per il font in uso.
Esempio:

```
p {font-size:60px; letter-spacing:50px}
```

- La proprietà **word-spacing**

Imposta la spaziatura tra le parole, in aggiunta allo spazio normale. Sono ammessi valori negativi. Con il valore **normal** la spaziatura è quella normale per il font in uso.

Esempio:

```
h1 {word-spacing: 0.5em}
```

- La proprietà **line-height**

Imposta l'iterlinea dell'elemento. Normalmente si usa un valore numerico senza unità di misura, per esempio 1, 1.2, 1.3, e così via, in modo che l'interlinea sia un mltiplicatore della dimensione del font.

Esempio:

```
p {line-height: 1.3}
interlinea pari a 1.3 volte la dimensione del font
```

- La proprietà **text-transform**

Questa proprietà controlla le maiuscole e le minuscole del testo di un elemento.
I valori sono:

capitalize - Trasforma il primo carattere di ciascuna parola in maiuscolo.
upper case - Trasforma tutti i caratteri di ciascuna parola in maiuscolo.
lowercase - Trasforma tutti i caratteri di ciascuna parola in minuscolo.
none - Nessuna trasformazione.

Esempio:

h1 {text-transform: capitalize}

Liste

- La proprietà **list-style-type**
  Definisce il tipo di marcatore di una lista.
  Gli attributi sono: **none**, **disc**, **circle**, **square**, **decimal**, **lower-roman**, **upper-roman**, **lower-alpha**, **upper-alpha**, **lower-greek** (non riconosciuto da tutti i browser).

  Esempio:
  ol {list-style-type:upper-alpha}

- La proprietà **list-style-image**
  Definisce un'immagine da usare come bullett in una lista.
  Esempio:
  ol {list-style-image: url(Cloud.gif)}

Margini e bordi

- Le proprietà **margin, margin-left, margin-right, margin-top, margin-bottom**.
  Definiscono la dimensione dei margini, tutti (margin), a sinistra (margin-left), a destra (margin-right), in alto (margin-top), in basso (margin-bottom)

  Esempio:
  body {margin:100px; }

- Le proprietà **padding, padding-left, padding-right, padding-top, padding-bottom**.
  Definiscono la dimensione del padding (distanza tra testo e margine), tutte (padding), a sinistra (padding-left), a destra (padding-right), in alto (padding-top), in basso (padding-bottom).

  Esempio:
  p {padding:100px; text-align:justify}

- La proprietà **border-style**.
  Imposta lo stile del bordo di un elemento, che può essere espresso con una delle seguenti parole chiave: **none**, **hidden**, **dotted**, **dashed**, **solid**, **double**, **groove**, **ridge**, **inset**, **outset**.

- La proprietà **border-color**.
  Imposta il colore del bordo di un elemento

- La proprietà **border-width**.
  Imposta lo spessore dei bordi; i valori possono essere: **thin**, **medium**, **thick** o un valore numerico.

- Le proprietà **border-top-width, border-bottom-width, border-left-width, border-right-width**.
  Impostano lo spessore, rispettivamente, del bordo superiore, inferiore, sinistro, destro.

Esempio:
p { border-style:dotted; border-color:blue; border-width:thick}
Ogni paragrafo compare all'interno di un riquadro a puntini spessi e di colore blue.

Tabelle

- **Il modello di tabella CSS**

Il *modello di tabella CSS* è basato sul modello di tabella HTML 4.0, detto "a supremazia di riga", poiché gli autori costruiscono le tabelle esplicitamente per righe. Il modello di tabella CSS consiste di tabelle (table), didascalie (caption), righe (tr), gruppi di riga (tbody), colonne (col), gruppi di colonna (colgroup) e celle (td, th).

- La proprietà **border-collapse**

Questa proprietà seleziona un modello di bordi per la tabella. Il valore **separate** imposta il **modello a bordi separati**, in cui le celle adiacenti sono separate tra di loro. Il valore **collapse**, invece, seleziona il **modello a bordi collassati**, in cui le celle adiacenti non sono separate tra di loro e i loro bordi si uniscono per formare un unico bordo.

Esempio:

```
table {border-collapse: collapse; border: 1px solid black;}
```

- La proprietà **border-spacing**

Quando si usa il modello a bordi separati, questa proprietà specifica la distanza che separa i bordi delle celle adiacenti. Se si specifica una sola lunghezza, questa imposta la spaziatura orizzontale e verticale. Con due lunghezze si impostano prima la spaziatura orizzontale e poi quella verticale. I valori non possono essere negativi.

Esempio:

```
table {border-collapse: separate; border: 1px solid #000;border-spacing: 3px 4px;}
```

Link

Anche lo stile dei collegamenti ipertestuali può essere personalizzato con i CSS.
Per la formattazione dei link vengono usati selettori speciali, detti **pseudo-classi**.
Le pseudo-classi, a differenza degli altri selettori, agiscono sullo stato di un elemento, cioè sono applicate solo se sono soddisfatte determinate condizioni. Per esempio, si può cambiare il colore di un link quando si passa su di esso con il mouse.

Le pseudo-classi sono precedute dal selettore, in questo caso "a" (il tag del link) e dal carattere ":", cui segue il nome della pseudo-classe.

Per i link sono:

- a:link si riferisce al link presente nella pagina e non ancora visitato;
- a:visited si riferisce al link già visitato;
- a:hover si riferisce al momento in cui il cursore del mouse passa sul link senza cliccare;
- a:active si riferisce al link sul quale si sta facendo clic (link attivo).

Esempio:

Barra di navigazione tale che, al passare del mouse sul link, i colori di primo piano e sfondo si invertano. La barra è all'interno di un contenitore .

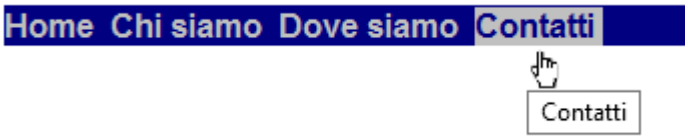
La pagina con la barra è:

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

```
        <link  rel="stylesheet" type="text/css" href="cssBarraNav.css">
        </head>
        <body>
           <div>
              <a href="index.htm" title="Home">Home </a>
              <a href="ChiSiamo.htm" title="Chi siamo">Chi siamo </a>
              <a href="DoveSiamo.htm" title="Dove siamo">Dove siamo </a>
              <a href="Contatti.htm" title="Contatti">Contatti </a>
           </div>
        </body>
     </html>
```

Il foglio di stile cssBarraNav.css è:

```
     div{
        background-color:navy;
     }

     a:link, a:visited {
        color:silver;
        text-decoration:none;
        font-size:1em;
        font-family:Arial;
        font-weight:bold;
     }

     a:hover {
        color:navy;
        background-color:silver;
     }
```



È possibile applicare le pseudo-classi anche ad altri elementi (paragrafi, immagini, tabelle, div, ecc.), producendo importanti effetti di animazione.

Esempi:

```
      p:hover {background-color:pink;
     passando  con il mouse sul paragrafo, lo sfondo diventa rosa;

      p:first-letter{font-size:40px;}
      la prima lettera del paragrafo ha dimensione 40px (si può ad esempio ingrandire la prima
      lettera);

      p:first-line{ background-color:pink;}
      la prima linea del paragrafo ha sfondo rosa;
```

Cursore

- La proprietà **cursor**

Questa proprietà specifica il tipo di cursore da visualizzare per il dispositivo di puntamento. I principali valori sono:

auto	Determinato dal browser in base al contesto.
crosshair	Una croce semplice (segno '+').
default	Dipendente dalla piattaforma. Spesso visualizzato come una freccia.
pointer	Un puntatore che indica un link.
move	Indica qualcosa che sta per essere mosso.
text	Indica il testo selezionabile. Spesso reso con una "I" maiuscola.
wait	Indica che il programma è occupato e l'utente deve aspettare. Spesso reso con un orologio o una clessidra.
progress	Indica l'avanzamento di stato del programma.
help	Indica un aiuto fornito per l'oggetto sotto il cursore. Spesso reso con un punto interrogativo o un fumetto.

Posizionamento

Le seguenti proprietà dei CSS permettono di specificare dove deve essere posizionato un elemento all'interno di una pagina html. Queste regole ormai sono diventate lo standard per creare un layout di una pagina web. Utilizzare i css per effettuare il layout di una pagina web presenta dei vantaggi rispetto ai layout mediante tabelle:

- pagine più corte in termini di linee di codice html;
- maggiore adattabilità ai diversi dispositivi di visualizzazione;
- caricamento delle pagine più veloce;
- maggiore visibilità da parte dei motori di ricerca.

Di default ogni elemento html viene posizionato secondo quello che viene detto **normale flusso del documento**. Secondo il normale flusso, gli elementi html possono essere di tipo:

- ***block-level* -** gli elementi vengono posizionati su una nuova riga , si estendono dall'alto al basso e tutto ciò che appare dopo di loro viene inserito in una nuova linea(Es. DIV, P, H1,...,H6, OL, UL, DL, HR);
- ***Inline* -** gli elementi appaiono all'interno della frase corrente, non partono da una nuova linea e si estendono dalla sinistra alla destra(Es. SPAN, B, I, U, IMG).

Per posizionare un elemento fuori dal normale flusso, si possono utilizzare due proprietà:

- ***position***
- ***float***

Proprietà position

Permette di definire come considerare lo spostamento di un elemento fuori dal normale flusso. Essa può assumere quattro valori

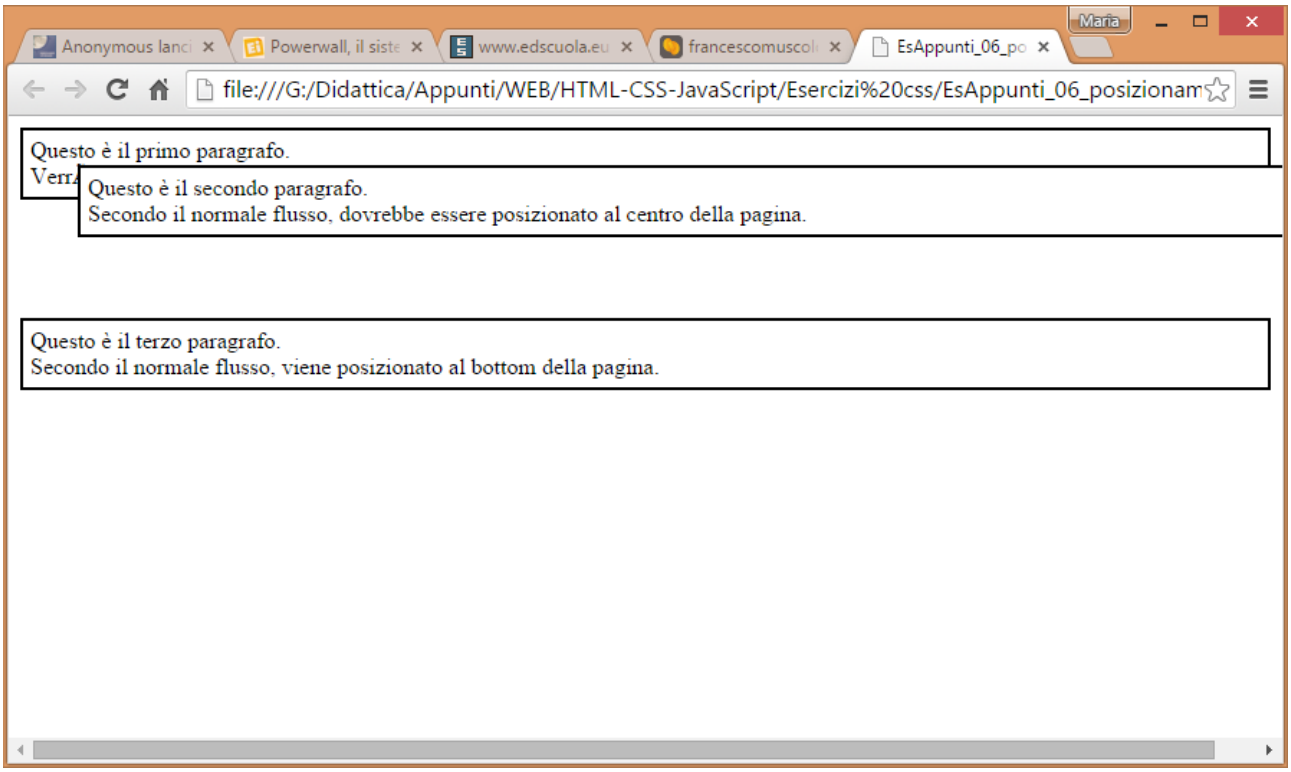
Valore	**significato**
static	È il valore di default e quindi indica che un elemento sarà posizionato secondo il normale flusso.
relative	Indica che l'elemento sarà posizionato specificando lo spostamento relativo al normale flusso.
absolute	Indica che l'elemento sarà posizionato specificando lo spostamento dall'angolo in alto a sinistra dell'elemento che lo contiene.
fixed	Indica che l'elemento sarà posizionato specificando lo spostamento dall'angolo in alto a sinistra della finestra del browser. Inoltre l'elemento non cambierà posizione nemmeno durante lo scroll window dell'utente.

Lo spostamento di un elemento può essere specificato usando le seguenti proprietà:

top	Spostamento dall'alto
left	Spostamento da sinistra
bottom	Spostamento dal fondo
Right	Spostamento da destra

Uno spostamento deve essere specificato utilizzando left o right e top o bottom.

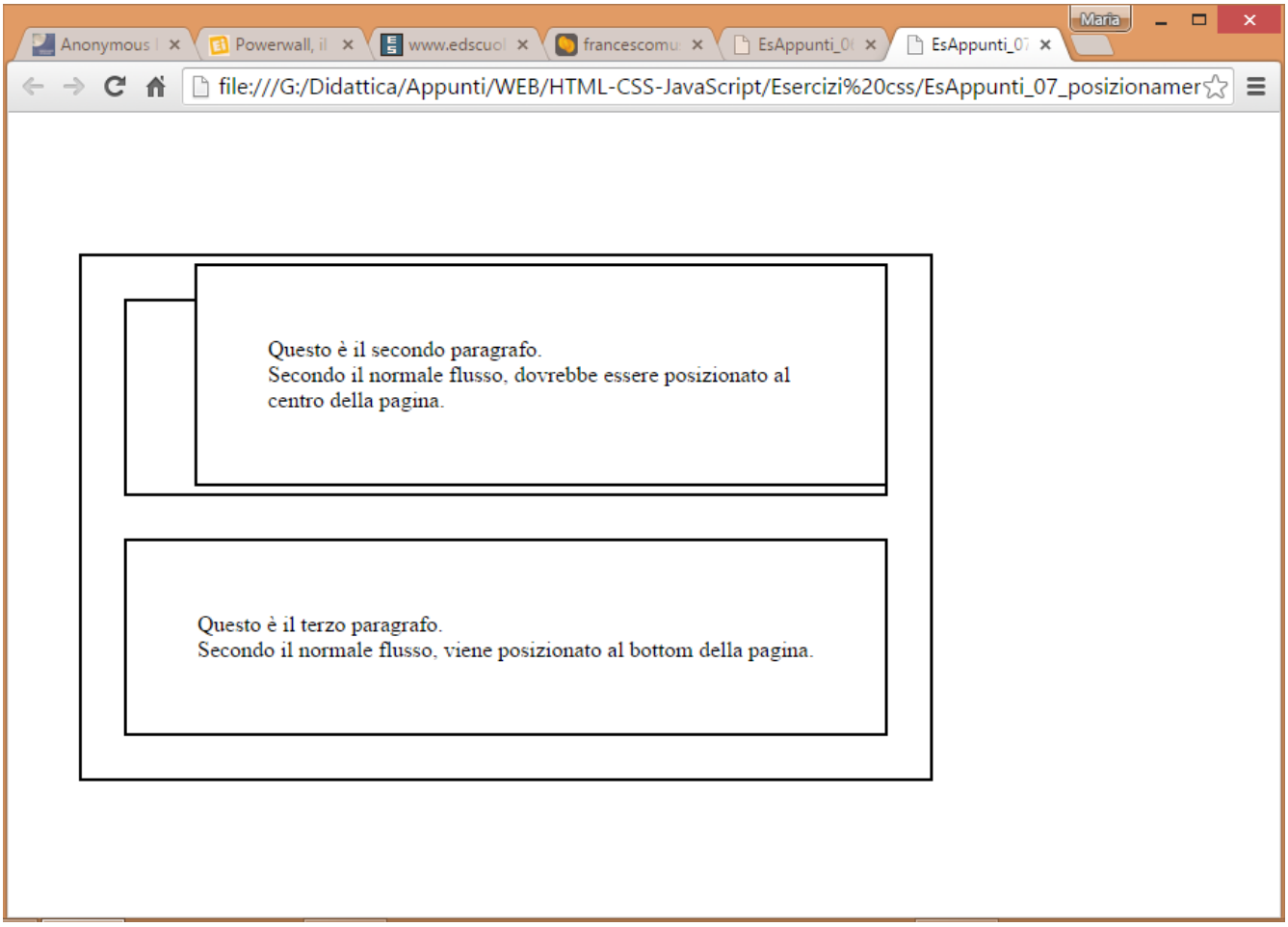
Esempio di position:relative




```
<html>
<head>
    <title></title>
    <style type="text/css">
        p {border-style:solid;
           border-color:#000000;
           border-width:2px;
           padding:5px;
           background-color:#FFFFFF;}
        p.due {
           position:relative;
           left: 40px;
           top: -40px;}
    </style>
</head>
<body>
     <p>
        Questo &egrave; il primo paragrafo. <br>Verrà posizionato al top della pagina.
     </p>
     <p class="due">
        Questo &egrave; il secondo paragrafo.<br>Secondo il normale flusso, dovrebbe
        essere posizionato al centro della pagina.
     </p>
     <p>
     Questo &egrave; il terzo paragrafo.<br>Secondo il normale flusso, viene
     posizionato al bottom della pagina.
```

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

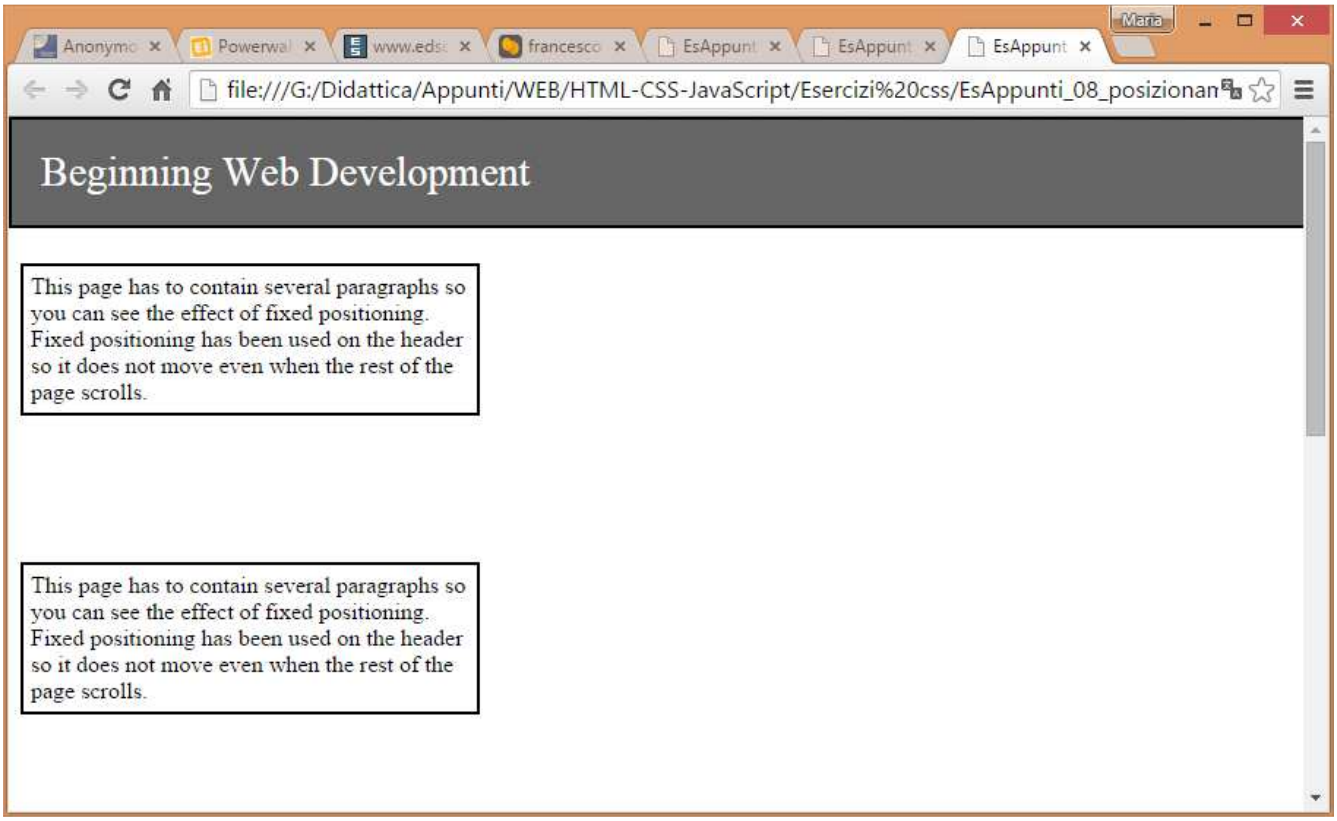
Esempio position:absolute




```
<html>
<head>
        <title></title>
        <style type="text/css">
                div.page {position:absolute;
                                left:50px;
                                top: 100px;
                                border-style:solid; border-width:2px; border-color:#000000;}
                p { background-color:#FFFFFF;
                    padding:50px;
                    border-style:solid; border-color:#000000; border-width:2px;
                    margin: 30px}
                p.due {position:absolute;
                        left:50px;
                        top: -25px;}
        </style>
</head>
<body>
  <div class="page">
      <p>
```

```
            Questo &egrave; il primo paragrafo. <br>Verr&agrave; posizionato al top della
            pagina
        </p>
        <p class="due">
            Questo &egrave; il secondo paragrafo.<br>Secondo il normale flusso, dovrebbe
            essere posizionato al centro della pagina.
        </p>
        <p>
            Questo &egrave; il terzo paragrafo.<br>Secondo il normale flusso, viene
            posizionato al bottom della pagina.
        </p>
    </div>
</body>
</html>
```

Esempio di position: fixed




```
<html>
<head>
    <title></title>
    <style type="text/css">
        div.intestazione
                { position:fixed;
                top: 0px;
                left:0px;
                width:100%;
                padding:20px;
                font-size:28px;
                color:#ffffff; background-color:#666666;
                border-style:solid; border-width:2px; border-color:#000000;}
        p { width:300px;
            padding:5px;
            color:#000000; background-color:#FFFFFF;
```

```
                    border-style:solid; border-color:#000000; border-width:2px;}
                p.uno {margin-top:100px; }
        </style>
</head>
<body>
<div class="intestazione"> Beginning Web Development </div>
<p class="uno"> This page has to contain several paragraphs so you can see
the effect of fixed positioning. Fixed positioning has been used on the
header so it does not move even when the rest of the page scrolls. </p>

<p class="uno"> This page has to contain several paragraphs so you can see
the effect of fixed positioning. Fixed positioning has been used on the
header so it does not move even when the rest of the page scrolls. </p>

<p class="uno"> This page has to contain several paragraphs so you can see
the effect of fixed positioning. Fixed positioning has been used on the
header so it does not move even when the rest of the page scrolls. </p>

<p class="uno"> This page has to contain several paragraphs so you can see
the effect of fixed positioning. Fixed positioning has been used on the
header so it does not move even when the rest of the page scrolls. </p>

<p class="uno"> This page has to contain several paragraphs so you can see
the effect of fixed positioning. Fixed positioning has been used on the
header so it does not move even when the rest of the page scrolls. </p>
</body>
</html>
```

Nel caso in cui più elementi si sovrappongano, si può utilizzare la proprietà **z-index** per specificare quale elemento deve apparire al top. Il valore di **z-index** è un numero. Apparirà al top, l'elemento che avrà il valore più grande per la proprietà **z-index**.

Proprietà float

La proprietà **float** è, insieme alla proprietà **clear**, la base su cui costruire layout basati sui CSS.
Con float è possibile rimuovere un elemento dal normale flusso del documento e spostarlo su uno dei lati (destro o sinistro) del suo elemento contenitore. Il contenuto che circonda l'elemento scorrerà intorno ad esso sul lato opposto rispetto a quello indicato come valore di float.
La sintassi generica è:

selettore {float: valore;}

I valori possibili sono:

left	l'elemento viene spostato sul lato sinistro del box contenitore, il contenuto scorre a destra.
right	l'elemento viene spostato sul lato destro, il contenuto scorre a sinistra.
none	valore iniziale e di default in mancanza di una dichiarazione esplicita; l'elemento mantiene la sua posizione normale.

Gli elementi float sono resi automaticamente **block-level**: questo significa che possono avere margini, bordi, padding e si può attribuire loro una larghezza e/o un altezza via CSS.
A differenza degli elementi block-level normali, gli elementi float, oltre ad essere **auto-adattanti** in altezza (ossia assumere l'altezza necessaria al contenuto), lo sono anche in larghezza. Questo significa che un elemento float, se non dichiarato attraverso dimensioni esplicite, assumerà la larghezza massima per il suo contenuto, fino a espandersi per tutta la larghezza del suo contenitore.

```
div {
 width: 200px;
 float: right;
}
img {float: left;}
```

È essenziale considerare che gli elementi float sono traslati dal flusso degli elementi di pagina verso uno dei due estremi del loro contenitore, e che elementi adiacenti nel codice "sentono" la loro presenza, regolandosi di conseguenza.
Una proprietà che accompagna spesso la proprietà float è la proprietà **clear** che, se applicata ad un elemento successivo ad uno reso float, impedisce che questo subisca il float.

Come detto, i float sono **auto-adattanti in larghezza**. Gli effetti di float non controllati in larghezza sono imprevedibili. Una buona pratica è quindi costringere il float in larghezza. I modi possibili sono due:

- attribuire una larghezza al suo contenuto
- usare esclusivamente contenuto di larghezza nota.

Nell'applicazione più semplice, la proprietà *float* può essere usata per allineare un'immagine efar scorrere testo intorno ad essa.
Esempio:

```
img {
   float: right;
   margin: 0 0 10px 10px;
}
```

Immagine allineata a destra. Il testo la affianca a sinistra.

Proprietà clear

La proprietà clear serve a impedire che al fianco di un elemento compaiano altri elementi con il float. Si applica solo agli elementi **blocco**.
L'origine di tale proprietà è questa: visto che il float sposta un elemento dal flusso normale del documento, è possibile che esso venga a trovarsi in posizioni non desiderate, magari al fianco di altri elementi che vogliamo invece tenere separati.
La sintassi è:

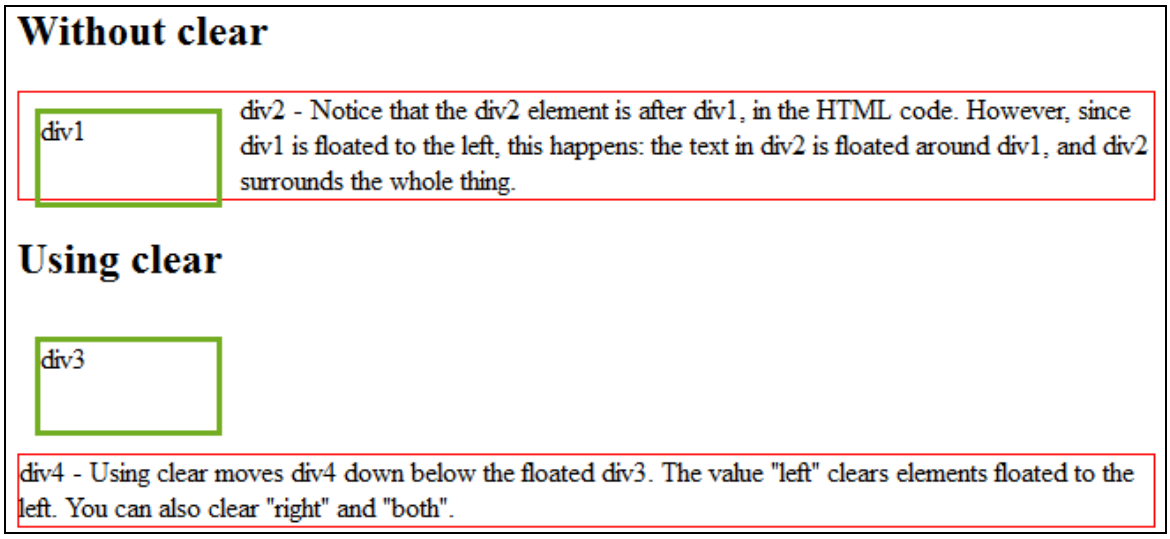
selettore {clear: valore;}

I valori possibili sono:

none	gli elementi con float possono stare a destra e sinistra dell'elemento;
left	si impedisce il posizionamento a sinistra;
right	si impedisce il posizionamento a destra;
both	si impedisce il posizionamento su entrambi i lati.

Esempio:

```
<!DOCTYPE html>
<html>
<head>
<style>
.div1 {float: left; width: 100px; height: 50px; margin: 10px;
      border: 3px solid #73AD21;}
.div2 {border: 1px solid red;}
.div3 {float: left; width: 100px; height: 50px; margin: 10px;
      border: 3px solid #73AD21;}
.div4 {border: 1px solid red; clear: left;}
</style>
</head>
<body>

<h2>Without clear</h2>
<div class="div1">div1</div>
<div class="div2">div2 - Notice that the div2 element is after div1, in the HTML
code. However, since div1 is floated to the left, this happens: the text in div2
is floated around div1, and div2 surrounds the whole thing.</div>

<h2>Using clear</h2>
<div class="div3">div3</div>
<div class="div4">div4 - Using clear moves div4 down below the floated div3. The
value "left" clears elements floated to the left. You can also clear "right" and
"both".</div>
```



Progettare una pagina web con i fogli di stile

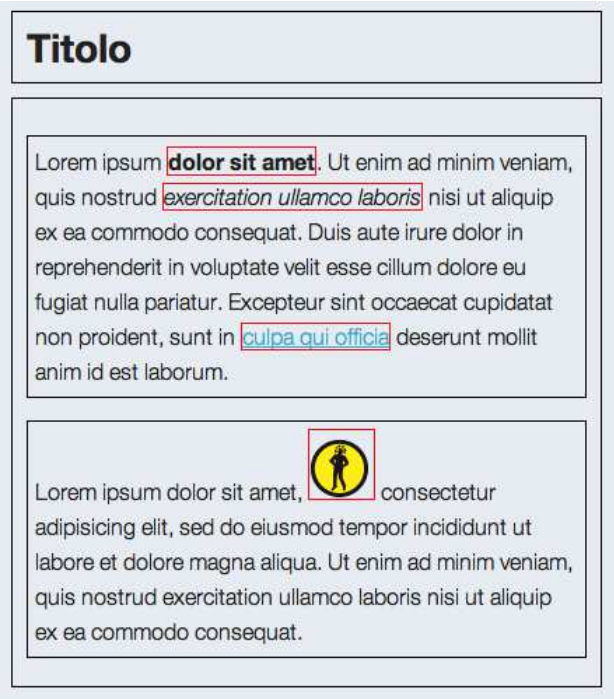
Un layout di pagina CSS utilizza i CSS al posto delle tradizionali tabelle o dei frame HTML, per organizzare il contenuto di una pagina Web. Il blocco costitutivo di base del layout CSS è il tag , in genere utilizzato come contenitore di testo, immagini e altri elementi della pagina.

Durante la creazione di un layout CSS, si possono inserire tag nella pagina, aggiungervi contenuti e collocarli in varie posizioni. A differenza delle celle di una tabella, le quali possono esistere solamente all'interno delle righe e delle colonne di una tabella, i tag possono apparire in qualsiasi punto di una pagina Web.

Si possono disporre i tag in modo assoluto (specificandone le coordinate x e y) o relativo (specificandone la posizione rispetto a quella corrente), o anche specificando float, spaziature e margini. È questo il metodo più diffuso in base agli attuali standard del Web design.

Elementi block-level ed elementi inline

Una pagina HTML, resa a schermo da un browser, è composta, di fatto, da un insieme di rettangoli (box). Non importa che si tratti di paragrafi, titoli, tabelle o immagini: si tratta sempre di box rettangolari. Alcuni box ne contengono altri al loro interno.



Come già detto, si fa distinzione tra elementi **block-level** (**a livello di blocco**), quelli che nella figura sono contrassegnati dal bordo nero, ed elementi **inline** (in linea), quelli circondati dal bordo rosso.

Un **elemento block-level** è un elemento che forma un blocco separato, può contenere altri elementi block-level e elementi inline e gli si possono attribuire delle dimensioni. Gli elementi block-level sono disposti verticalmente (al termine di un blocco si va a capo), formando per ciascuno una nuova riga. Un elemento block level di dimensioni non specificate occupa tra margini, bordi, padding e contenuto, **tutta la larghezza messa a disposizione del suo box**

contenitore. In verticale occuperà l'altezza necessaria al suo contenuto.

Sono esempi di elementi block-level:
- **div**, che è un blocco contenitore (div è abbreviazione di "DIVisore")
- **frameset**, contenitore di frames multipli
- **h1**, **h2**, **h3**, **h4**, **h5**, **h6**
- **hr**
- **p**
- **table** (i suoi elementi "figli" si comportano come elementi a livello di blocco e può contenere altre tabelle)
- **form**

Anche i seguenti elementi possono essere considerati a livello di blocco, dal momento che possono contenere elementi a livello di blocco:
- **dd**
- **dt**
- **li**
- **td**
- **th**
- **tr**

Tra di essi, questi sono denominati **elementi List-item**:
- **dl**
- **li**
- **ol**
- **ul**

Un elemento inline è un elemento che rimane in linea con il resto del contenuto: può contenere solo altri elementi inline. A un elemento inline, a meno che questo non venga dichiarato float, posizionato o modificandone la sua natura con la proprietà display, **non si possono attribuire dimensioni**. Elementi inline adiacenti sono disposti ***orizzontalmente***. Un elemento inline occuperà, sia in orizzontale sia in verticale, lo spazio necessario al suo contenuto.

Sono esempi di elementi inline:
- **a**
- **br**
- **b**, **i**, **u**
- **img** (immagine con altezza e larghezza intrinseca)
- **span** (è un contenitore per testo da formattare in modo speciale, non produce un "a capo" a termine del testo)
- **sub**
- **sup**
- **textarea**

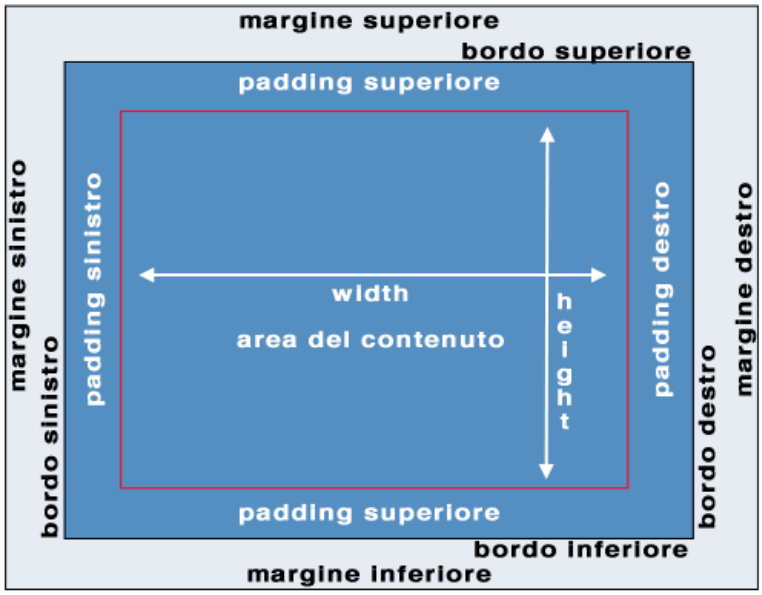
Esempio:

```
<h1>Titolo</h1>
<p>Paragrafo</p>
```

Le parole "titolo" e "paragrafo" appariranno su due righe diverse, perché <h1> e <p> sono elementi blocco.

Il box model

Si tratta del meccanismo che governa la presentazione dei vari elementi di una pagina. Come detto, infatti, una pagina HTML non è altro che un insieme di box rettangolari, che si tratti di elementi **block-level** o di elementi **inline**.
Ogni box comprende un certo numero di componenti di base, ciascuno modificabile con proprietà dei CSS. La figura qui sotto mostra visivamente tali componenti:



Partendo dall'interno abbiamo:

- **L'area del contenuto**: è la zona in cui trova spazio il contenuto vero e proprio: testo, immagini, video, etc.; le dimensioni orizzontali dell'area possono essere modificate con la proprietà **width**, quelle verticali con **height**.
- **Il padding**: è uno spazio vuoto che può essere creato tra l'area del contenuto e il bordo dell'elemento; come si vede nella figura, se si imposta un **colore di sfondo** per un elemento, esso si estende dall'area del contenuto alla zona di padding.
- **Il bordo**: è una linea di dimensione, stile e colore variabili, che circonda la zona del padding e l'area del contenuto.
- **Il margine**: è uno spazio di dimensioni variabili che separa un dato elemento da quelli adiacenti.

Riguardo alla larghezza del box, si fa distinzione tra:

- larghezza dell'area del contenuto;
- larghezza complessiva;
- larghezza dell'area visibile.

La prima è data dal valore della proprietà **width**.

La seconda rappresenta lo spazio occupato sulla pagina, compresi i margini, ed è data dalla somma:

margine sinistro + bordo sinistro + padding sinistro
+ area del contenuto
+ padding destro + bordo destro + margine destro

La terza corrisponde allo spazio occupato sulla pagina, esclusi i margini, vale a dire la parte del box delimitata dai bordi. È data da questa somma:

bordo sinistro + padding sinistro
+ area del contenuto
+ padding destro + bordo destro

Esempio:

```
div {
 width: 200px;
 padding-left: 10px;
 padding-right: 10px;
 border-left: 5px solid black;
 border-right: 5px solid black;
}
```

Un div così definito non occuperà sulla pagina una larghezza pari a 200px. Il valore di **width** fa riferimento solo all'area del contenuto. Per calcolare la larghezza reale ed effettiva, bisognerà aggiungere ai 200px i valori per padding e bordi.
Quindi:

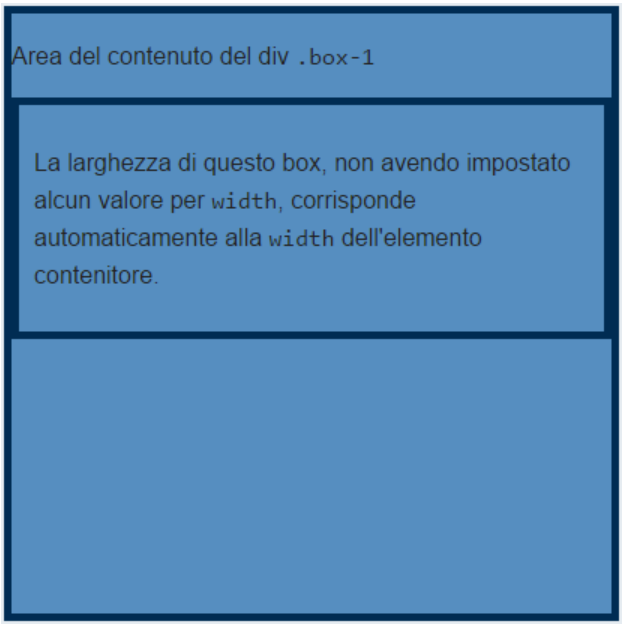
10 + 5 + 200 + 10 + 5, ovvero: 230px.

Il valore per **width** può essere

- **auto**: valore iniziale e di default; se non si impostano margini, bordi e padding la larghezza dell'elemento sarà uguale all'area del contenuto dell'elemento contenitore;
- **un valore numerico con unità di misura**;
- **un valore in percentuale**: la larghezza sarà calcolata rispetto a quella dell'elemento contenitore.

Se non si imposta un valore per la proprietà **width** o se si usa **auto**, la larghezza di un box è uguale a quella dell'area del contenuto dell'elemento contenitore. Quest'ultimo è l'elemento che racchiude il box.

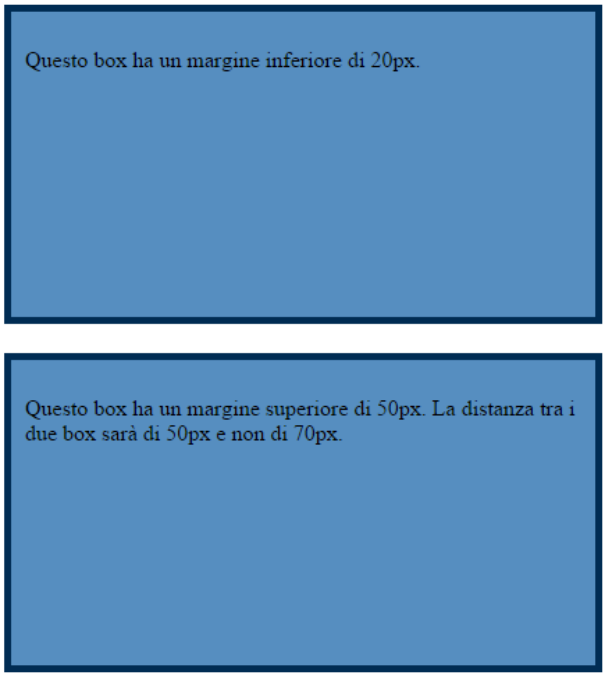
Esempio: Larghezza in base al contenitore.




```
<html lang="it">
<head>
<meta charset="UTF-8" />
<title></title>
<style type="text/css">
        .box-1 {
         width: 400px;
         height: 400px;
         border: 5px solid #002c53;
         background: #568ec0;
        }
        .box-2 {
         border: 5px solid #002c53;
         background: #568ec0;
        }
</style>
</head>
<body>
        <div class="box-1">
                <p>Area del contenuto del div .box-1</p>
```

```
        <div class="box-2">
                <p> La larghezza di questo box, non avendo impostato alcun valore per width,
                        corrisponde automaticamente alla width dell'elemento contenitore</p>
        </div>
        </div>
</body>
</html>
```

Per due box adiacenti in senso verticale che abbiano impostato un margine inferiore e uno superiore, la distanza non sarà data dalla somma delle due distanze. A prevalere sarà invece la distanza maggiore tra le due. È il meccanismo del cosiddetto **margin collapsing**. Tale meccanismo non si applica ai box adiacenti in senso orizzontale.



In genere **l'altezza di un elemento è determinata dal suo contenuto**. Più testo s'inserisce in un box, più esso sarà esteso in senso verticale. La **proprietà height** definisce la distanza tra il bordo superiore e quello inferiore di un elemento.
Sintassi:

selettore {height: valore;}

Il valore può essere espresso da:

- **un valore numerico con unità di misura**;
- **un valore in percentuale**: il valore in percentuale è sempre definito rispetto all'altezza del blocco contenitore, purché esso abbia un'altezza esplicitamente dichiarata; diversamente, la percentuale viene interpretata come auto;
- **auto**: l'altezza sarà quella determinata dal contenuto.

Esempi:

```
div {height: 250px;}
ul {height: 50%;}
p {height: auto;}
```

La **proprietà overflow** è strettamente collegata alla proprietà height. Essa fornisce un modo per gestire il contenuto che superi i limiti imposti con height. Serve infatti per definire il comportamento di un elemento blocco nel caso il suo contenuto ecceda dalle sue dimensioni esplicite, ossia nel caso in cui l'area utile del box non sia sufficiente per i contenuti. È da usare quando si vogliano creare box ad altezza fissa ma con molti contenuti.
Sintassi:

selettore {overflow : valore;}

I valori possono essere espressi con le parole chiave:

- **visible**: valore iniziale, il contenuto eccedente rimane visibile;
- **hidden**: il contenuto eccedente non viene mostrato;
- **scroll**: il browser crea barre di scorrimento che consentono di fruire del contenuto eccedente;
- **auto**: il browser tratta il contenuto eccedente secondo le sue impostazioni predefinite; di norma dovrebbe mostrare una barra di scorrimento laterale.

Esempi:

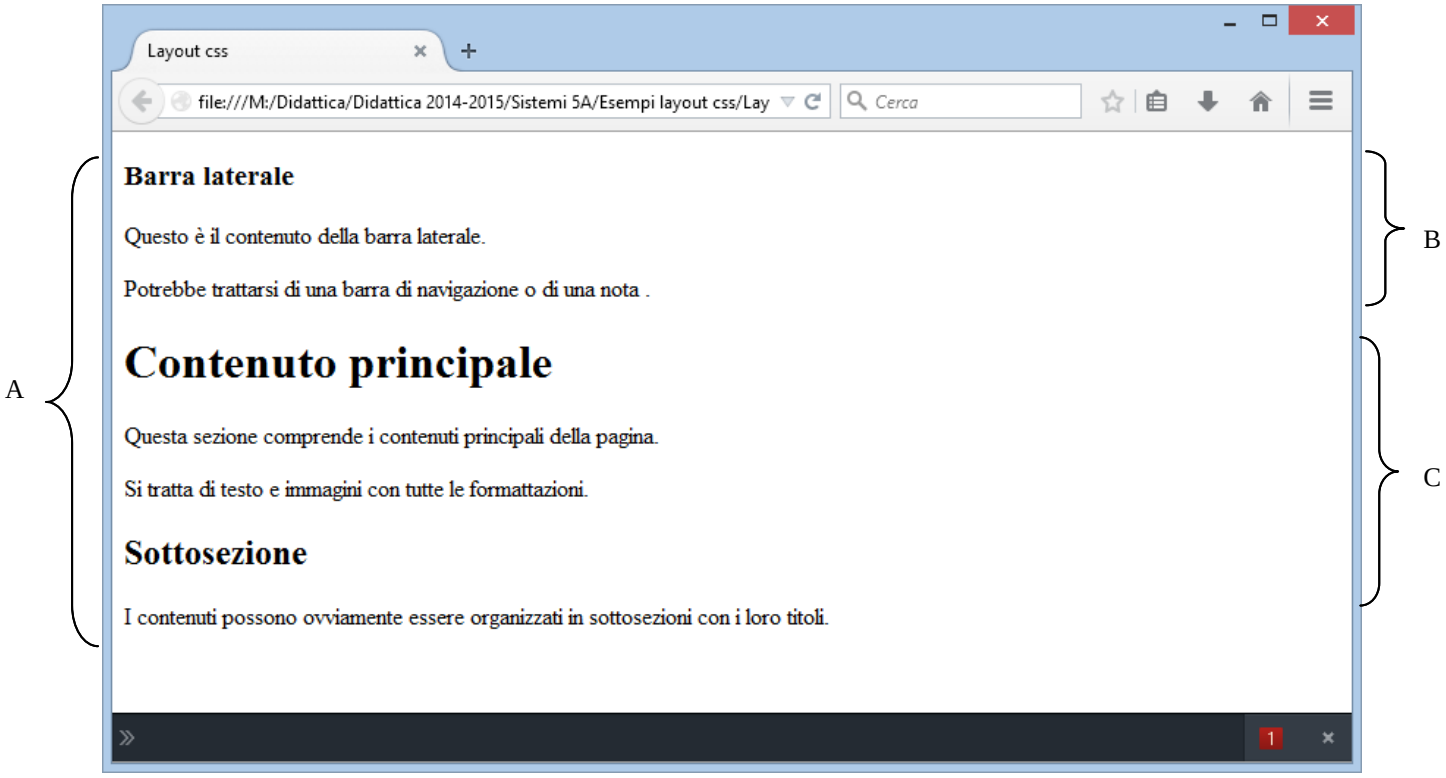
```
div {overflow: auto;}
p {overflow: hidden;}
div {overflow: visible;}
p {overflow: scroll;}
```

L'attributo **margin** accetta anche come valore **auto**, che posizionerebbe il blocco al centro del flusso visto che assegnerebbe automaticamente lo stesso margine ad entrambi i lati. Ad essere centrato è il blocco e non il suo contenuto. Attenzione anche al fatto che, per avere un bordo nel foglio di stile, non basta specificare il solo spessore (*1px*), si deve specificare anche un possibile attributo che identifica il tipo di bordo usato, in questo caso *solid*, diversamente il bordo non risulterà visibile.

Struttura dei layout di pagina CSS

Vediamo ora, attraverso un esempio, come si può impostare il layout di una pagina HTML utilizzando i fogli di stile.
La pagina in figura è strutturata in tre tag separati:
- tag "contenitore" di grandi dimensioni (A)
- barra laterale (B)
- contenuto principale della pagina (C).

A contiene B e C.



A. div contenitore - B. div barra laterale - C. div contenuto principale

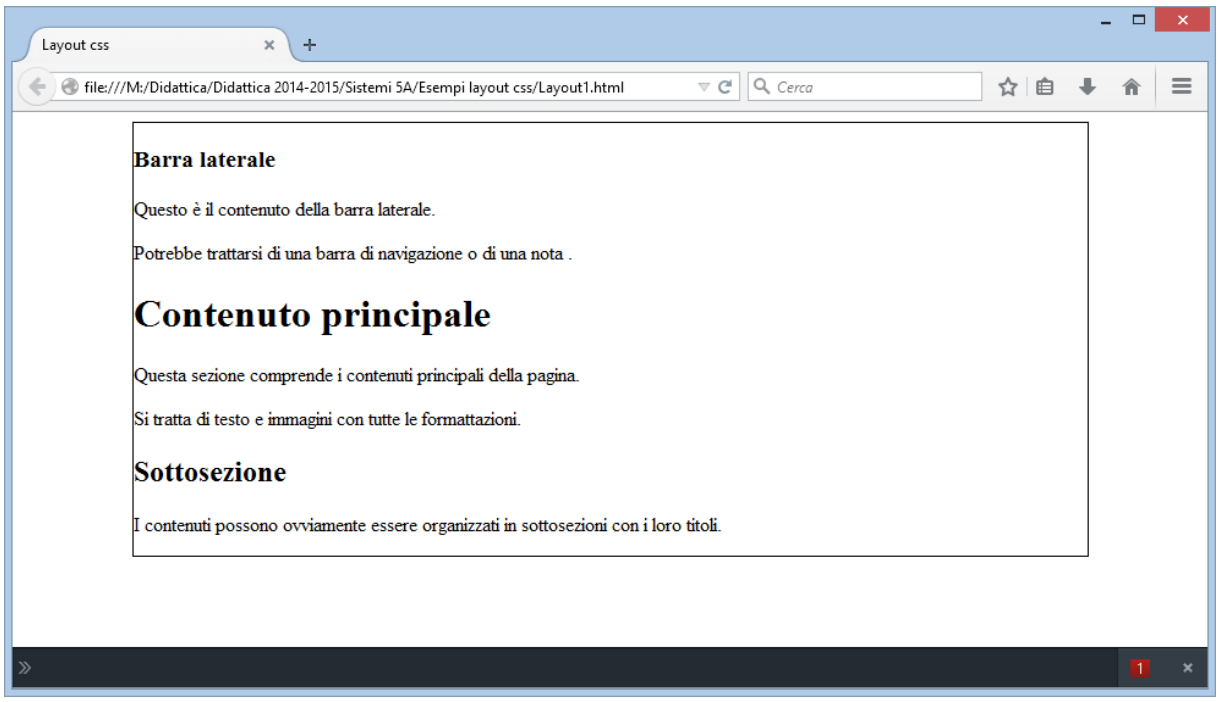
Il codice che segue crea i tre tag nella pagina HTML:

```
<html>
<head><title>Layout css</title></head>
<body>
   <div id="contenitore">
        <div id="barralaterale">
              <h3>Barra laterale</h3>
              <p>Questo &egrave; il contenuto della barra laterale.</p>
              <p>Potrebbe trattarsi di una barra di navigazione o di una nota .</p>
        </div>
        <div id="contenutoprincipale">
              <h1>Contenuto principale</h1>
              <p>Questa sezione comprende i contenuti principali della pagina.</p>
              <p>Si tratta di testo e immagini con tutte le formattazioni.</p>
              <h2>Sottosezione </h2>
              <p>I contenuti possono ovviamente essere organizzati in sottosezioni
              con i loro titoli.</p>
        </div>
   </div>
</body>
</html>
```

Nell'esempio, ai tag non è applicato alcuno stile. Senza alcuna regola CSS definita, i tag e i relativi contenuti sono inseriti in una posizione predefinita nella pagina, determinata dalla sequenza con cui sono riportati.
Tuttavia, se ciascun tag dispone di un ID univoco, tali ID possono essere impiegati per creare regole CSS che, quando applicate, ne modificano lo stile e il posizionamento.
La regola CSS seguente, che può essere inserita nella sezione head del documento o in un file CSS esterno, crea regole di stile per il primo tag "contenitore" della pagina:

```
#contenitore {
   width: 780px;
   background: #FFFFFF;
   margin: 0 auto;
   border: 1px solid #000000;
   text-align: left; }
```

I risultati dell'applicazione di questa regola al tag div contenitore sono i seguenti:



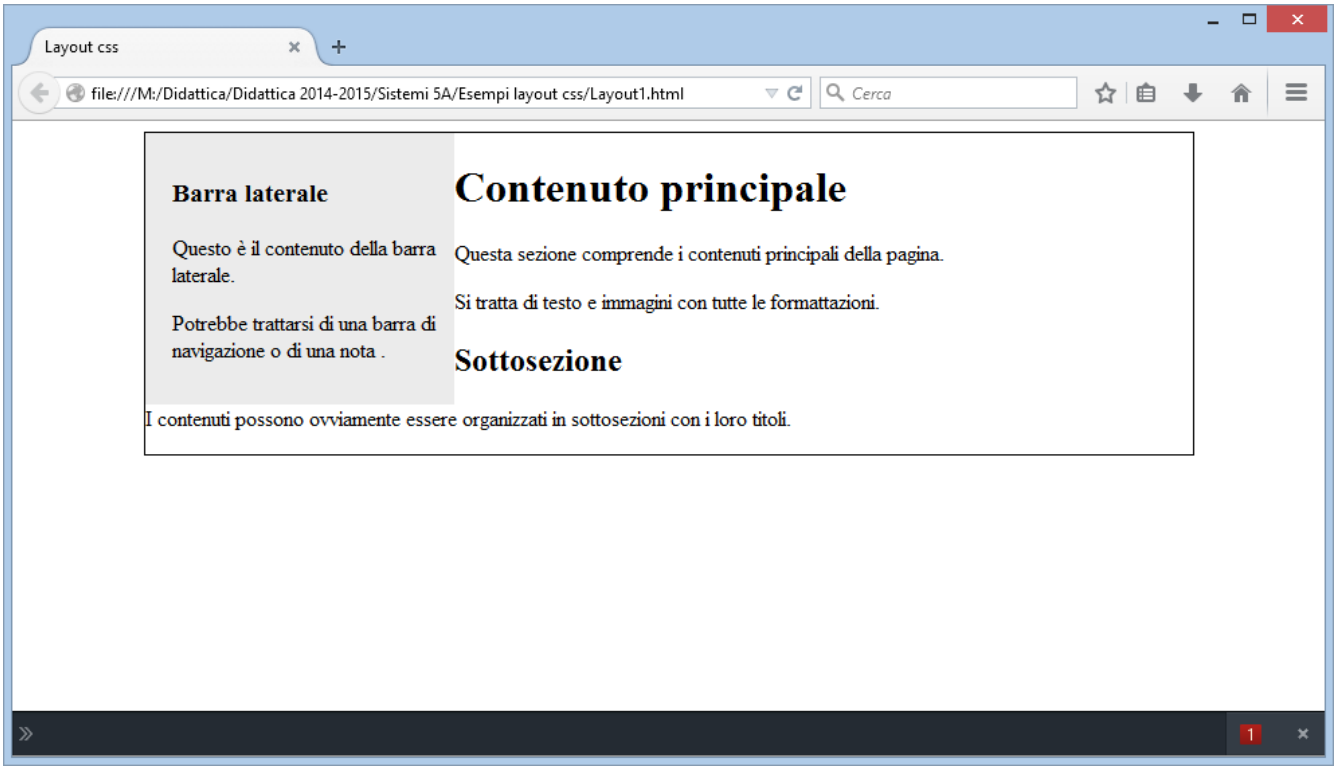
La regola *#contenitore* definisce lo stile del tag contenitore, in modo da assegnargli:

- larghezza di 780 pixel,
- sfondo di colore bianco,
- nessun margine dal lato sinistro della pagina,
- bordo solido di colore nero di un pixel
- testo allineato a sinistra.

La regola CSS successiva crea regole di stile per il tag usato per la barra laterale:

```
#barralaterale {
   float: left;
   width: 200px;
   background: #EBEBEB;
   padding: 15px 10px 15px 20px;}
```

I risultati dell'applicazione di questa regola al tag per la barra laterale sono i seguenti:



La regola #barralaterale definisce lo stile del tag in modo da assegnargli:

- larghezza di 200 pixel,
- sfondo di colore grigio,
- spaziatura superiore e inferiore di 15 pixel,
- spaziatura destra di 10 pixel,
- spaziatura sinistra di 20 pixel.
  (l'ordine predefinito per la spaziatura è alto-destra-basso-sinistra)

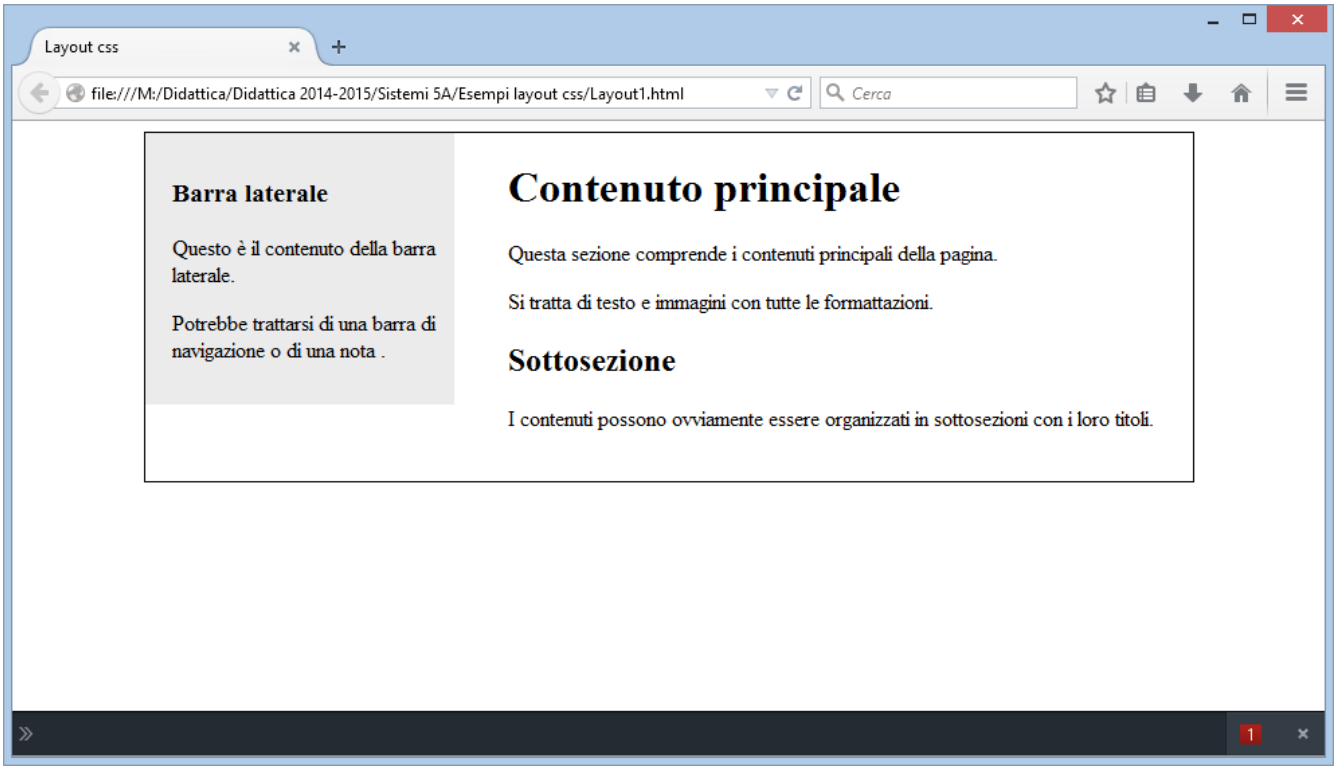
La regola posiziona il tag con la proprietà **float: left**, che allinea il tag per la barra laterale al lato sinistro del tag del contenitore.

Infine, la regola CSS per il tag con il contenuto principale completa il layout:

```
#contenutoprincipale {
margin: 0 0 0 250px;
padding: 0 20px 20px 20px;}
```

La regola *#contenutoprincipale* definisce lo stile del tag per il contenuto principale con un margine sinistro di 250 pixel, inserendo quindi uno spazio di 250 pixel tra il lato sinistro del tag contenitore e il lato sinistro del tag del contenuto principale. Inoltre, la regola prevede 20 pixel di spazio a destra, sotto e a sinistra del tag del contenuto principale. I risultati dell'applicazione di questa regola al tag div *#contenutoprincipale* sono i seguenti:

La pagina, con il codice completo, ha il seguente aspetto:

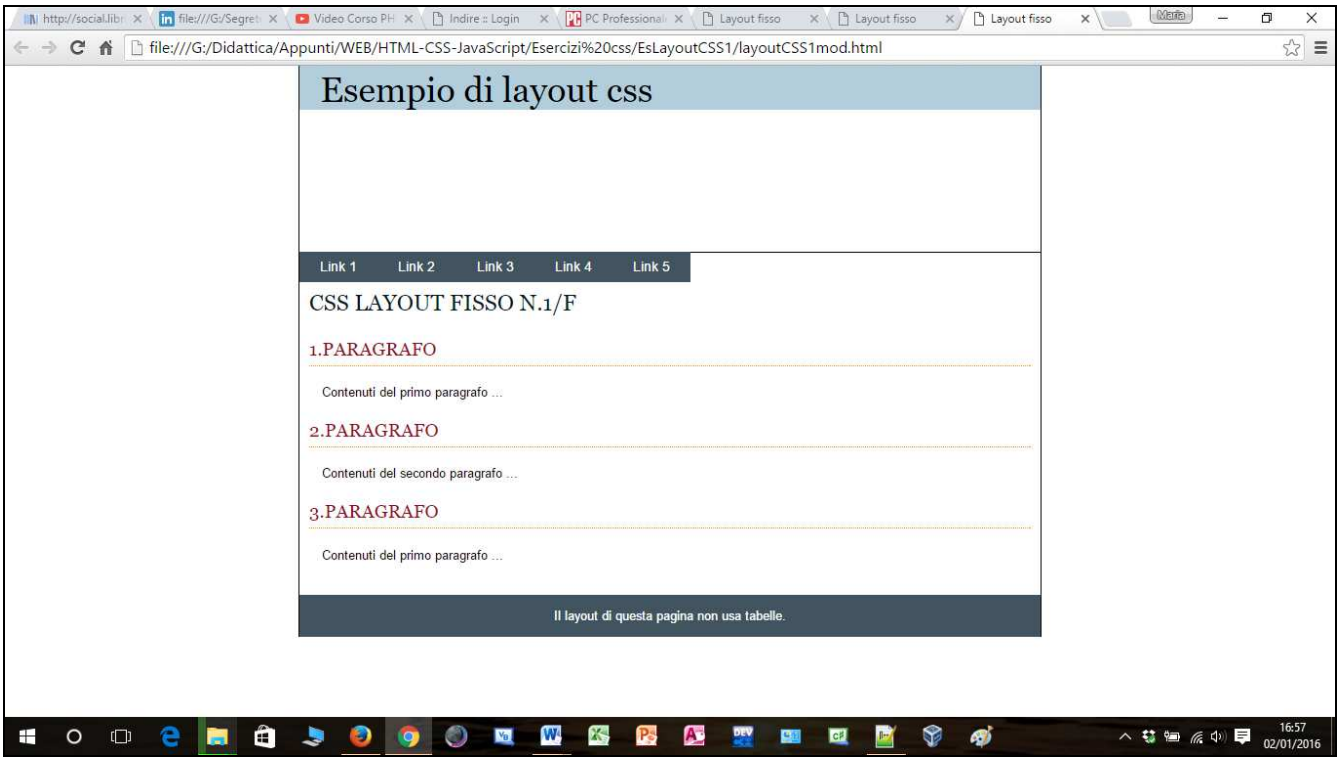


Ricapitolando:

```
<html>

<head>
        <title>Layout css</title>
        <style type="text/css">
                #contenitore {
                width: 780px;
                background: #FFFFFF;
                margin: 0 auto;
                border: 1px solid #000000;
                text-align: left; }

                #barralaterale {
                float: left;
                width: 200px;
                background: #EBEBEB;
                padding: 15px 10px 15px 20px;}

                #contenutoprincipale {
                margin: 0 0 0 250px;
                padding: 0 20px 20px 20px;}
        </style>
</head>
```

```
<body>
	<div id="contenitore">
		<div id="barralaterale">
			<h3>Barra laterale</h3>
			<p>Questo &egrave; il contenuto della barra laterale.</p>
			<p>Potrebbe trattarsi di una barra di navigazione o di una nota .</p>
		</div>
		<div id="contenutoprincipale">
			<h1>Contenuto principale</h1>
			<p>Questa sezione comprende i contenuti principali della pagina.</p>
			<p>Si tratta di testo e immagini con tutte le formattazioni.</p>
			<h2>Sottosezione </h2>
			<p>I contenuti possono ovviamente essere organizzati in sottosezioni
				con i loro titoli.</p>
		</div>
	</div>
</body>

</html>
```

Esempio:



EsempioLayout.html

```
<head>
<title>Layout fisso </title>
<link href="default01f.css" rel="stylesheet" type="text/css" title="default" />
</head>
<body>
	<div id="header">
	<h1>Esempio di layout css</h1>
	</div>
	<div id="menu">
```

```
                        <ul>
                        <li><a href="#" title="link 1">Link 1</a></li>
                        <li><a href="#" title="link 2">Link 2</a></li>
                        <li><a href="#" title="link 3">Link 3</a></li>
                        <li><a href="#" title="link 4">Link 4</a></li>
                        <li><a href="#" title="link 5">Link 5</a></li>
                        </ul>
        </div>
<div id="content">
        <h1>CSS LAYOUT FISSO N.1/F</h1>
        <h2>1.PARAGRAFO</h2>
        <p>    Contenuti del primo paragrafo ...</p>
        <h2>2.PARAGRAFO</h2>
        <p>Contenuti del secondo paragrafo ...</p>
        <h2>3.PARAGRAFO</h2>
        <p>Contenuti del primo paragrafo ... </p>
</div>
<div id="footer">Il layout di questa pagina non usa tabelle.<br /></div>
</body>
</html>
```

default01f.css

```
body{
        margin: 0;
        padding: 0;
        font-family: arial, helvetica, verdana, tahoma, sans-serif;
        font-size: 80%;
        color: #000;
        background-color: #fff;
        line-height: 180%;
}
#header{
        margin: 0 auto;
        width: 760px;
        height: 190px;
        border: 1px solid #000;
        border-top: 1px;
        border-bottom: 1px;
        background-image: url(img01.jpg);
        background-repeat: no-repeat;
        background-position: right bottom;
        color: #000;
}
#header h1{
        margin: 0;
        font: normal 280% Georgia, "Times New Roman", Times, serif;
        color: #000;
        background-color: #B2CEDC;
        padding-left: 0.6em;
        padding-top: 0.1em;
}
#menu{
        margin: 0 auto;
        width: 760px;
        background: #41545F;
        border-right: 1px solid #000;
        border-left: 1px solid #000;
        border-bottom: 1px solid #000;
        color: #fff;
```

```
}
#menu ul{
	margin: 0;
	padding: 0;
}
#menu li{
	font-size: 1.1em;
	display:inline;
	margin: 0;
	padding: 0;
}
#menu a:link, #menu a:visited{
	float: left;
	background: #41545F;
	color: #fff;
	margin: 0em;
	padding: 0.3em 1.5em 0.3em 1.5em;
	text-decoration: none;

}
#menu a:hover, #menu a:active{
	color: #B2CEDC;
	background: #05181E;
}
#content{
	margin: 0 auto;
	width: 740px;
	padding: 1.25em 0.8em;
	border: 1px solid #000;
	border-top: 0;
	border-bottom: 0;
	background: #fff;
	color: #000;
}
#content p {
	font-size: 100%;
	line-height: 1.8em;
	padding-left: 1em;
	padding-right: 1em;
}
#content h1 {
	padding-bottom: 0.3em;
	padding-top: 0.3em;
	font: normal 180% Georgia, "Times New Roman", Times, serif;
	color: #05181E;
	background-color: #FFFFFF;
}
#content h2{
	background: #fff;
	color: #940D1E;
	padding-bottom: 0.3em;
	font: normal 150% Georgia, "Times New Roman", Times, serif;
	border-bottom: 1px dotted #FF9006;
}

#footer{
	margin: 0 auto;
	width: 740px;
	background: #41545F;
	text-align: center;
```

```
        color: #fff;
        border: 1px solid #000;
        border-top: 1px;
        border-bottom: 1px;
        font-family: helvetica, arial, verdana, tahoma, sans-serif;
        padding: 0.8em 0.8em;
}
```

Il Box Liquido

Abbiamo visto come si creano i box e come si posizionano correttamente.
Esiste però anche una tecnica, che prende il nome di **Box Liquido**, che consente al box di adattarsi alla pagina, ridimensionandosi automaticamente, proprio come fosse un liquido versato in un recipiente che ne assume la forma.

Una semplice soluzione consiste nel posizionare un box al centro della pagina, impostando il valore **auto** alla proprietà **margin**

```
#box-centrato{
        margin: auto;
}
<div id="box-centrato">
        Contenuto del box
</div>
```

Esempio:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
      "http://www.w3.org/TR/html4/loose.dtd">
<html lang="it">
<head>
        <meta charset="UTF-8" />
        <title>Guida Fogli di Stile CSS | Esempio Box Liquido</title>
        <style type="text/css">
           body {
           padding: 0;
           margin: 0;
           }
           #box-centrato {
           margin: auto;
           color: black;
           background-color: #dfefdf;
           font-family: Verdana, Arial, sans-serif;
           font-size: 11pt;
           border: solid 1px #d9cfb0;
           }
           strong {color: #ff0000; font-weight: normal;}
           em {color: #0099ff;}
        </style>
</head>
<body>
        <div id="box-centrato">Questo box è centrato orizzontalmente, se provate a
        ridimensionare la finestra del browser il box ed il suo contenuto cambieranno dimensioni
        adattandosi perfettamente a quelle della nuova finestra.<br><br>
        Non ci sono spazi fra il box e la finestra del browser perchè <strong>margin</strong> e
        <strong>padding</strong> sono stati impostati a <em>0</em> nel body.
        </div>
</body>
</html>
```

A caratterizzare un box liquido è però fondamentalmente l'uso di valori percentuali per le proprietà che ne definiscono le caratteristiche (margin, padding, ecc.).

I Layout fissi e liquidi

I concetti di layout fisso e layout liquido da sempre contraddistinguono altrettante scuole di pensiero relative alla costruzione dei layout per siti web. Lunghe discussioni sono state fatte per stabilire pregi e difetti di queste tecniche e per cercare di distinguere il modo migliore di procedere; nonostante tutto, esistono tuttora pareri discordanti.

I **layout fissi** sono sostanziamente quelli che adottano come unità di misura il pixel. Mediante questa tecnica si ha la certezza che le pagine saranno rese sempre nello stesso modo, indipendentemente dalla caratteristiche tecniche delle macchine che le visualizzano. Un layout fisso avrà sempre le stesse dimensioni, anche se il dispositivo utilizzato dall'utente avrà dimensioni più piccole.

I **layout fluidi** invece sono quelli in cui tutte le misure sono espresse in forma percentuale. Mediante questa tecnica è possibile a costruire layout che si adatteranno sempre alle misure dello schermo dell'utente. Anche questo approccio però non è privo di controindicazioni: in schermi molto larghi, infatti, la lettura di righe lunghissime potrebbe rivelarsi problematica; un layout eccessivamente allargato risuta in genere essere sgradevole alla vista. Rappresentare, inoltre, le dimensioni in percentuali non mette al sicuro da processi di arrotondamento differenti da browser a browser che potrebbero causare una resa non omogenea del sito.

Ricapitolando:

Vantaggi del layout fluido
- la pagina si adatterà sempre alla grandezza dello schermo e del browser dell'utente.

Svantaggi del layout fluido
- è impossibile gestire al meglio la grafica in modo omogeneo;
- con i nuovi monitor, sempre più grandi, il layout rischia di ridursi ad una riga di testo illeggibile;
- i testi contenuti nei box possono risultare meno leggibili poichè distribuiti su poche righe di larghezza.

Vantaggi del layout fisso
- la grafica, così come viene progettata, rimarrà sempre la stessa ad ogni risoluzione;
- il testo risulta più leggibile, poichè contenuto nelle gabbie originariamente pensate per esso.

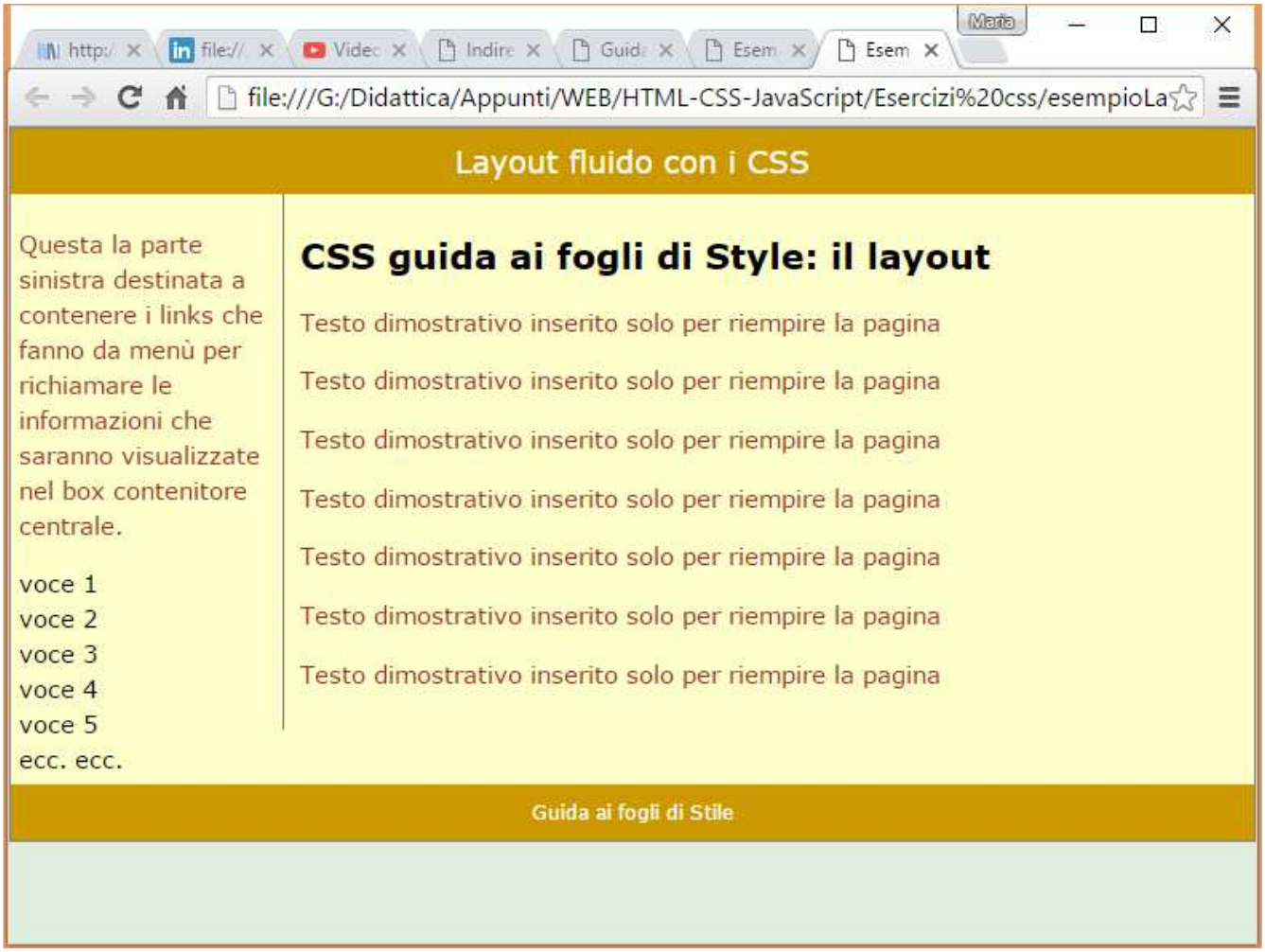
Svantaggi del layout fisso
- la grafica, proprio perchè fissa, a grandi risoluzioni apparirà "vuota" ai lati.

Esempio di layout fluido.

```
<!doctype HTML>
<html lang="it">
<head>
        <meta charset="UTF-8" />
        <title>Esempio di Layout fluido </title>

        <style type="text/css">

                body {
                        background-color: #dfefdf;
                        font-family: Verdana, Arial, sans-serif;
                        color: #000066;
                        padding: 0;
                        margin: 0;
                        font-size: 11pt;
                }

                #telaio {
                        border:1px solid gray;
                        line-height:150%;
                        color: black;
                        background-color: #ffc;
                }

                #testata{
                        font-size: 130%;
                }

                #fondo {
                        font-size: 85%;
                        clear: both;
                }

                #testata, #fondo {
                        padding:0.5em;
                        color:white;
                        background-color: #c90;
                        text-align: center;
                }
```

```
                #sinistro {
                        float:left;
                        width:160px;
                        margin: 0px;
                        padding: 5px;
                }

                #centrale {
                        margin-left: 170px;
                        border-left:1px solid gray;
                        padding:10px;
                }

                p.cs1{
                        color: brown;
                        font-size:14;
                        font-family: Verdana
                }
        </style>
</head>

<body>

<div id="telaio">

        <div id="testata">
          Layout fluido con i CSS
        </div>

        <div id="sinistro">
                <p class="cs1">Questa la parte sinistra destinata a contenere i links che fanno da
                menù per richiamare le informazioni che saranno visualizzate nel box contenitore
                centrale.
                </p>
                voce 1<br>
                voce 2<br>
                voce 3<br>
                voce 4<br>
                voce 5<br>
                ecc. ecc.
        </div>

        <div id="centrale">
                <h2>CSS guida ai fogli di Style: il layout</h2>
                <p class="cs1">Testo dimostrativo inserito solo per riempire la pagina </p>
                <p class="cs1">Testo dimostrativo inserito solo per riempire la pagina </p>
                <p class="cs1">Testo dimostrativo inserito solo per riempire la pagina </p>
                <p class="cs1">Testo dimostrativo inserito solo per riempire la pagina </p>
                <p class="cs1">Testo dimostrativo inserito solo per riempire la pagina </p>
                <p class="cs1">Testo dimostrativo inserito solo per riempire la pagina </p>
                <p class="cs1">Testo dimostrativo inserito solo per riempire la pagina </p>
        </div>

        <div id="fondo">
          Guida ai fogli di Stile</div>
        </div>
</div>
</body>
</html>
```

Questo il risultato a video:



JAVA SCRIPT

Cos'è JavaScript e a cosa serve

In una pagina web, oltre ai tag HTML, si possono incorporare sequenze di comandi, che prendono il nome di **script** e possono essere eseguiti durante la visualizzazione della pagina, in genere al verificarsi di eventi.
Javascript è un linguaggio di scripting web, il primo a essere apparso, molto diffuso agli inizi della sua vita ma anche adesso. È un linguaggio lato client, per cui gli script sono interpretati direttamente dal browser, a differenza dei linguaggi lato server.
E' stato sviluppato da Netscape Communications Corporation. Il nome iniziale non era questo, bensì Mocha, poi divenne LiveScript ma è stato cambiato nuovamente e in maniera definitiva in JavaScript perché la sintassi si basa su quella di Java (e quindi di C), comunque non bisogna confonderli perché sono due linguaggi differenti.
JavaScript è un linguaggio interpretato, perché il browser esegue il codice script riga per riga, a differenza dei linguaggi compilati, che richiedono la traduzione nel codice macchina. Per questo motivo si parla di script e non di programma.
Le principali applicazioni di JavaScript sono:

- fornire messaggi di avviso all'interno di una pagina di un sito;
- eseguire operazioni matematiche o su stringhe (ad esempio, conteggio di lettere o parole);
- gestire gli eventi che possono accadere durante la consultazione delle pagine, provocati dall'utente con i clic o i movimenti del mouse (ad esempio, le immagini animate e cambiare la loro visualizzazione quando si passa il mouse sopra di esse;
- controllare la validità di dati inseriti in un form;
- identificare il tipo di browser in utilizzo e visualizzare un contenuto adatto e diverso per ogni browser;
- ricercare i plugin installati e avvisare l'utente se è richiesto un ulteriore plugin per continuare.

Caratteristiche del linguaggio

JavaScript è un linguaggio **basato su oggetti**, ma non totalmente OOP, perché non supporta le classi. È anche **pilotato dagli eventi**.

Gli oggetti di JS sono:

Window
Document
Location
Navigator
History
Form
Ancore
Link
Array
String
Date
Math

più tutti quelli eventualmente definiti dall'utente.

Ogni oggetto è caratterizzato da un insieme di **proprietà** [p] che ne specificano le caratteristiche (nome, dimensioni, etc.), a richiesta può compiere determinate azioni o **metodi** [m] (aprire una finestra, inviare dei dati, etc.), può reagire a determinati **eventi** [e] esterni (click del mouse, pressione di un tasto, etc.) eseguendo dei metodi.

Per far riferimento a una proprietà o a un metodo, si scrive il nome dell'oggetto, seguito da un punto e dal nome della proprietà o del metodo. Il nome dell'oggetto in molti casi si può omettere. A differenza dei linguaggi C, C++ e Java, non è obbligatorio concludere le iscrizioni con il simbolo punto e virgola.

Inserimento degli script nelle pagine HTML

Uno script JavaScript può essere inserito
- all'interno della pagina HTML,
- oppure caricato da un file esterno (come file ASCII) con estensione .js.

Nel primo caso l'inizio e la fine del codice sono individuati rispettivamente dai tag **<script>** e **</script>**, usando l'attributo type o l'attributo language per specificare il nome del linguaggio:

```
<script type="text/javascript">
...
</script>
```

oppure

```
<script language="text/javascript">
...
</script>
```

Se lo script è all'interno del documento, può essere immesso sia nella sezione di intestazione (tra i tag <head></head>) sia nel corpo del documento (tra i tag <body></body>).
Il codice è eseguito in ordine sequenziale: dall'alto in basso, per cui lo script nell'intestazione viene eseguito prima degli altri, quello nella sezione body, invece, viene eseguito nella posizione in cui si trova.
Nel caso in cui il codice sia memorizzato in un file esterno, la sintassi nella pagina HTML è:

```
<script type="text/javascript" src=" nomefile.js">
 </script>
```

Commenti

All'interno di uno script i commenti vengono posti all'interno dei simboli: /* e */.
Il commento può essere posto su più righe o su una riga singola.
Un altro tipo di commento è la doppia barra (//), ma è valida solo per commenti posti su una singola riga.

Variabili

In Javascript le variabili sono dichiarate facendo precedere il loro nome dalla parola chiave **var**:

var NomeVariabile

All'atto della dichiarazione, è possibile anche assegnare un valore iniziale. Un esempio di inizializzazione e assegnamento è il seguente:

var NomeVariabile=Espressione

Esempio:

var x=10

La dichiarazione non è obbligatoria e può comparire in una qualsiasi posizione all'interno del codice. Per maggiore chiarezza, è però opportuno disporle in testa al codice.

La dichiarazione var su più variabili non va ripetuta per ognuna di esse, ma occorre separare i nomi delle variabili con una virgola.

Esempio:

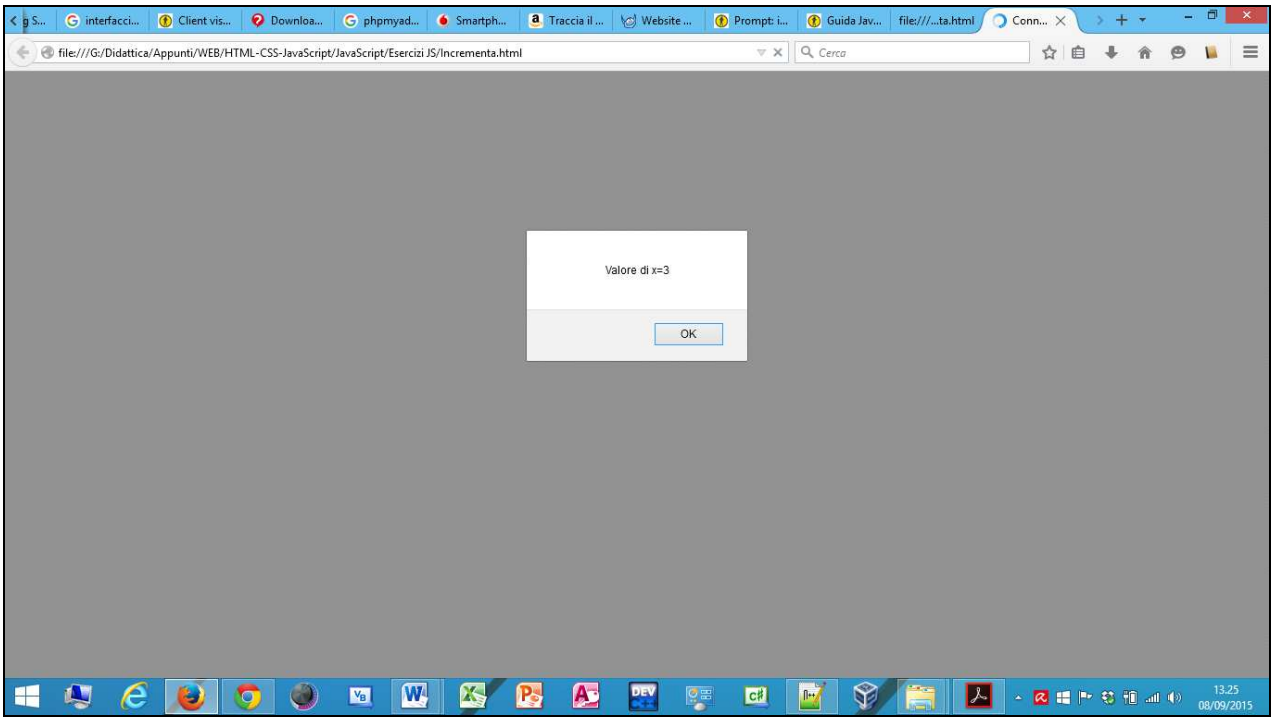
var x,y,z

Esempio:

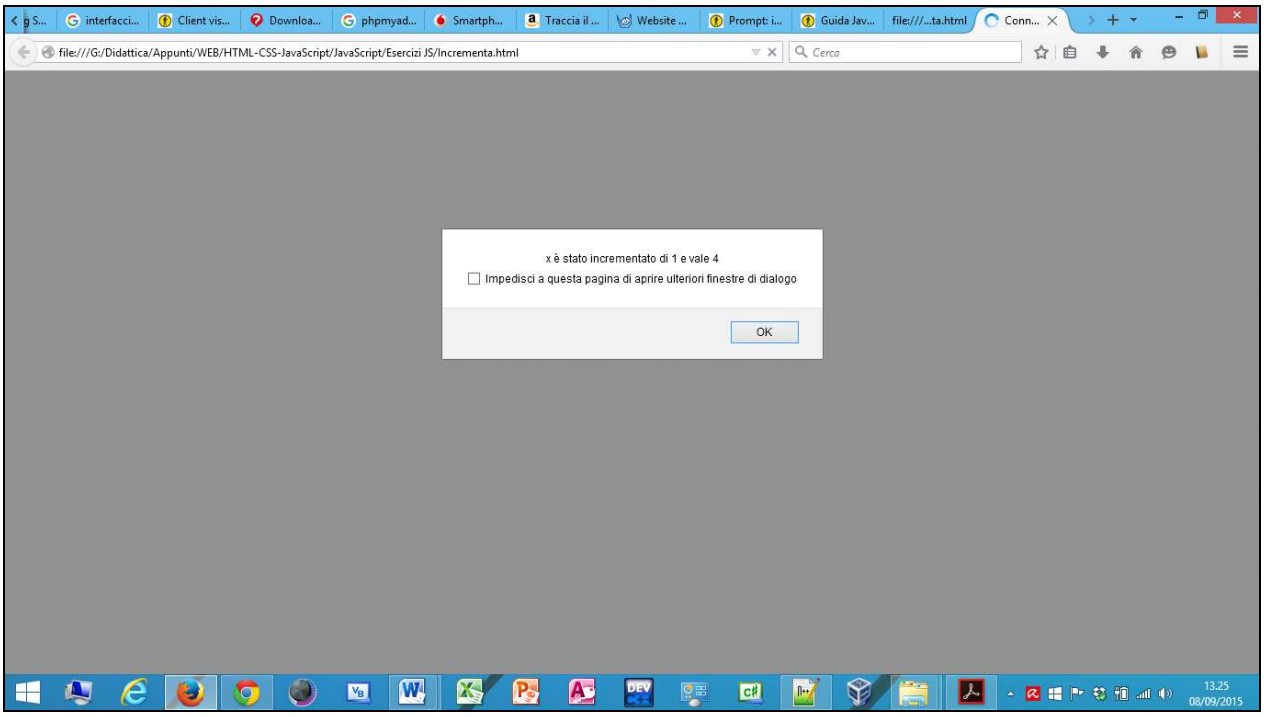
```
<!doctype HTML>
<html lang="it">

<head>
<meta charset="UTF-8" />
<script type="text/javascript">
      var x,y
      y=1
      x=y+2
      alert ("Valore di x="+x)
</script>
</head>

<body>
<script language="Javascript">
   x=x+y
   alert ("x è stato incrementato di "+y+" e vale " +x)
</script>
</body>

</html>
```

- alert è una finestra di dialogo che visualizza il messaggio tra parentesi tonde.
- + è l'operatore di concatenazione di stringhe.

Quando la pagina è caricata, compare:



Cliccando su OK:



Si può anche inizializzare una variabile con il valore **null** (var variabile=null).
Questo valore può apparire poco importante, ma diventa essenziale quando si vuole verificare se una variabile è stata caricata.

Le variabili che sono dichiarate senza essere inizializzate assumono valore **undefined**.
Ad esempio, dopo la dichiarazione:

var Verifica

la variabile Verifica assume il valore undefined.

È importante sapere che:

- i nomi delle variabili possono contenere solo lettere, numeri e trattino di sottolineatura, per cui sono esclusi gli spazi bianchi;
- il primo carattere del nome di una variabile deve essere sempre una lettera;
- Javascript è case sensitive, per cui tratta diversamente le lettere in maiuscolo e in minuscolo;
- nei nomi delle variabili, non si possono utilizzare le parole chiave del linguaggio.

Finestre di dialogo

I messaggi di avviso, altresì detti finestre di dialogo, sono una utility messa a disposizione degli sviluppatori da Javascript, di fatto si tratta di finestre che si aprono in popup al determinarsi di un evento o di una condizione.

Abbiamo a disposizione tre differenti tipi di finestre di messaggio:

- **alert** – finestra di avviso che comunica un messaggio; il dialogo avviene solo dal sito all'utente;
- **confirm** – finestra di scelta che comunica un messaggio e richiede una conferma all'utente, per cui il dialogo è bilaterale;
- **prompt** – finestra che richiede l'immissione di un dato, per cui il dialogo avviene dall'utente al sito.

Si tratta di tre metodi dell'oggetto **window**, anche se non è necessario scriverli comprendendo il nome dell'oggetto di riferimento.

Il metodo **alert()** visualizza in una output-box (una piccola finestra predefinita) una stringa o il valore di una variabile. La sintassi è:

window.alert (Stringa)

oppure

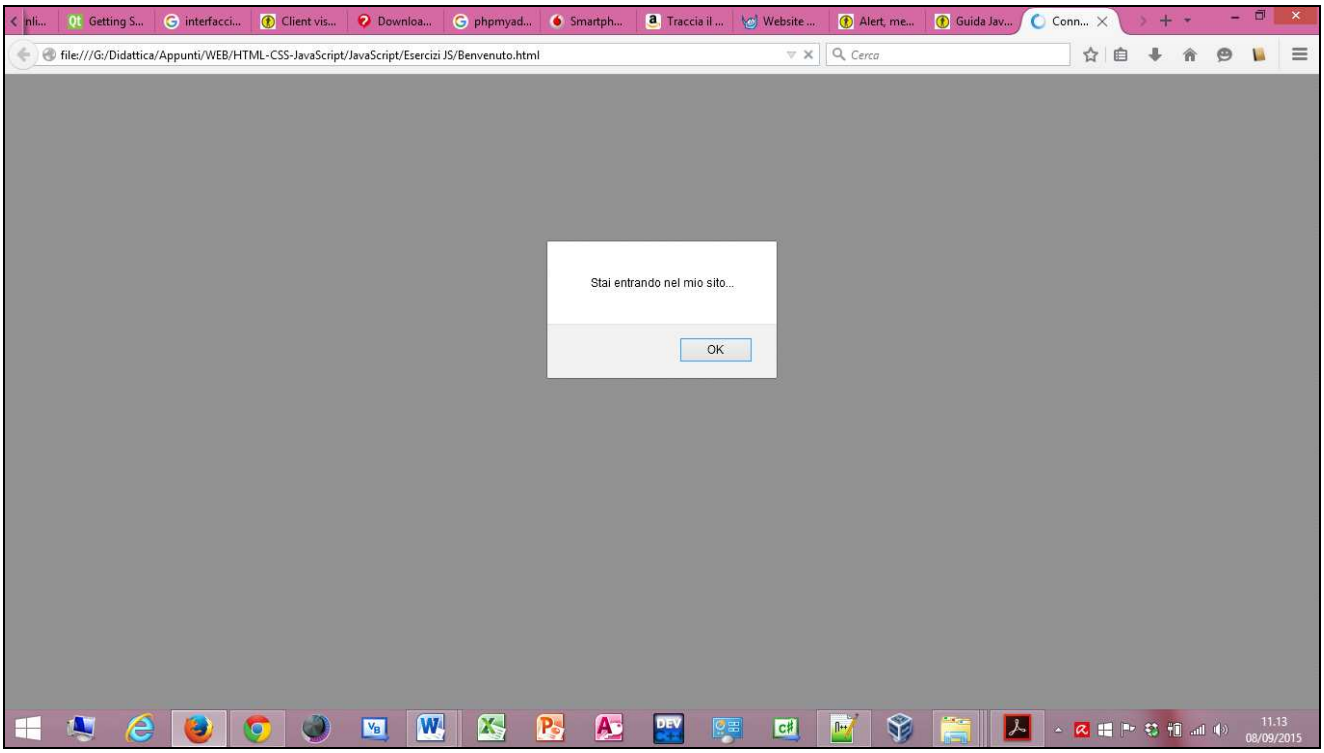
alert (Stringa)

dove *Stringa* può essere
- una costante stringa racchiusa tra doppi apici;
- una variabile;
- una concatenazione di costanti stringhe e variabili; l'operatore di concatenazione è +.

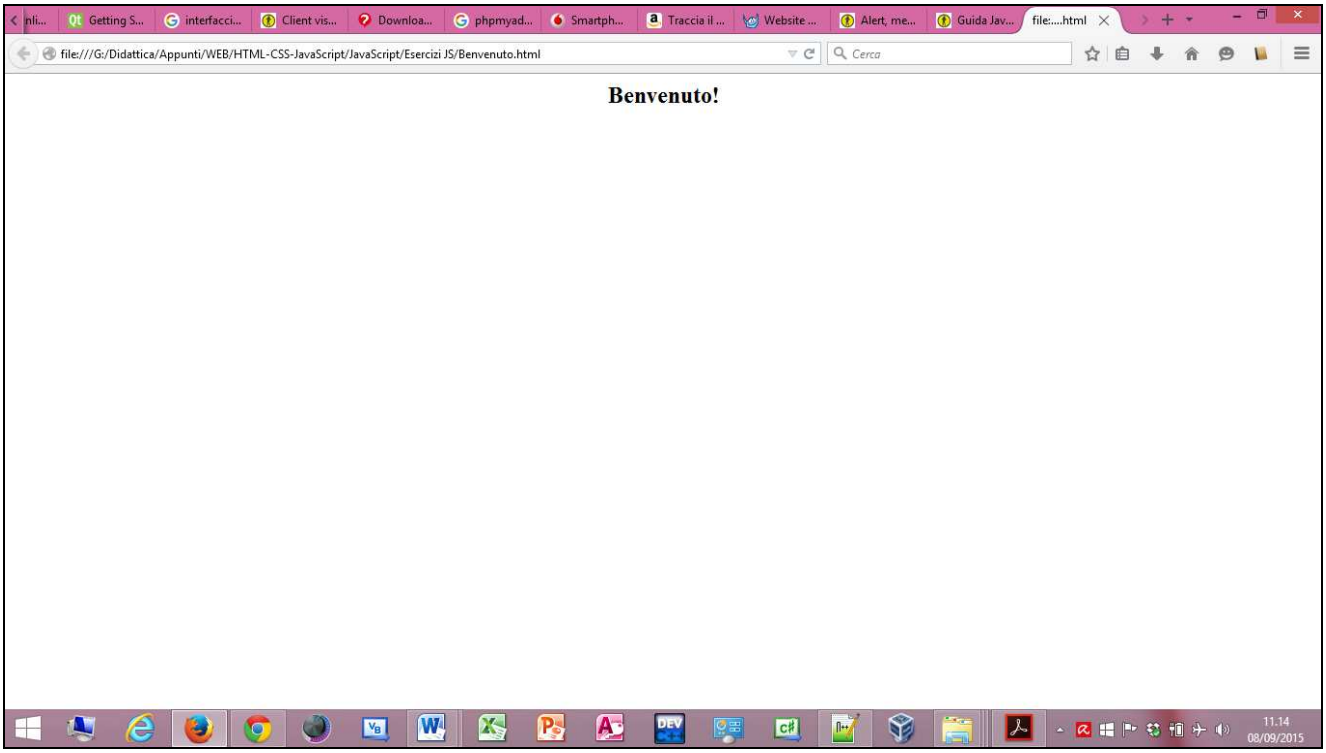
Esempio

```
<!doctype HTML>

<html lang="it">

<head>
    <meta charset="UTF-8" />
    <script language="javascript">
        window.alert("Stai entrando nel mio sito...")
    </script>
</head>

<body >
    <h2 align="center">Benvenuto!</h2>
</body>

</html>
```

Essendo lo script inserito nella sezione head, va in esecuzione appena la pagina è caricata, per cui compare subito:



Cliccando su OK:

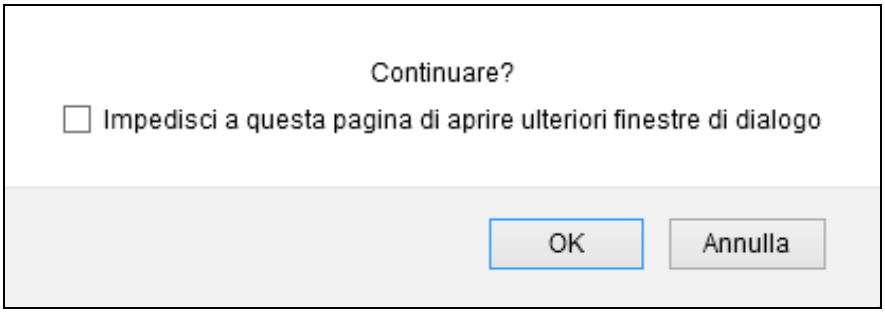


L'immagine della finestra dipende dal browser e dalle impostazioni del sistema operativo.

Il metodo **confirm()** scrive su un'output-box la stringa passata come parametro, ma, rispetto al metodo alert, propone in più una scelta tramite due pulsanti: OK e Cancel(Annulla).
Se l'utente preme OK, confirm restituisce true, altrimenti restituisce false, per questo è in genere richiamato con l'assegnazione a una variabile o in un test.
L'utilizzo più classico di questa finestra prende forma con il classico tasto "Annulla operazione" presente nei moduli da compilare.

Esempio:

var richiesta = window.confirm("Reimpostare il modulo o continuare?");



È necessario poi controllare il valore della variabile richiesta per eseguire le operazioni opportune.

Per far sì che l'utente possa invece inserire un valore da attribuire a una variabile, si può utilizzare il metodo **prompt()**, con la sintassi:

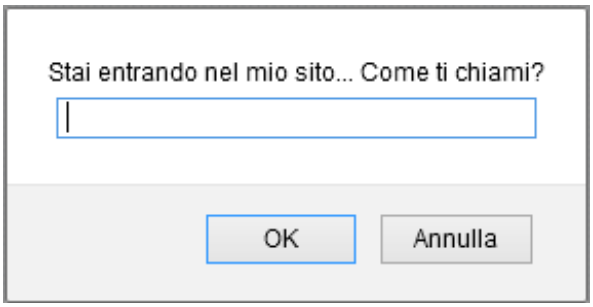
NomeVariabile=prompt("Messaggio", ValorePreimpostato)

dove:

- NomeVariabile è la variabile cui sarà assegnato il valore inserito dall'utente;
- Messaggio è una stringa che serve a richiedere il valore che l'utente dovrà inserire;
- ValorePreimpostato è un valore da far comparire nella casella di testo quando la finestra comparirà, come suggerimento all'utente, che comunque può cancellarlo o modificarlo.

Esempio:

var Nome=window.prompt("Stai entrando nel mio sito... Come ti chiami?")



Se si devono acquisire valori reali o interi si devono utilizzare rispettivamente le funzioni parseFloat e parseInt per le opportune conversioni.

Es.
x=parseInt(prompt("Inserire un numero intero"))
y=parseFloat(prompt("Inserire un numero reale"))

Scrittura nella pagina

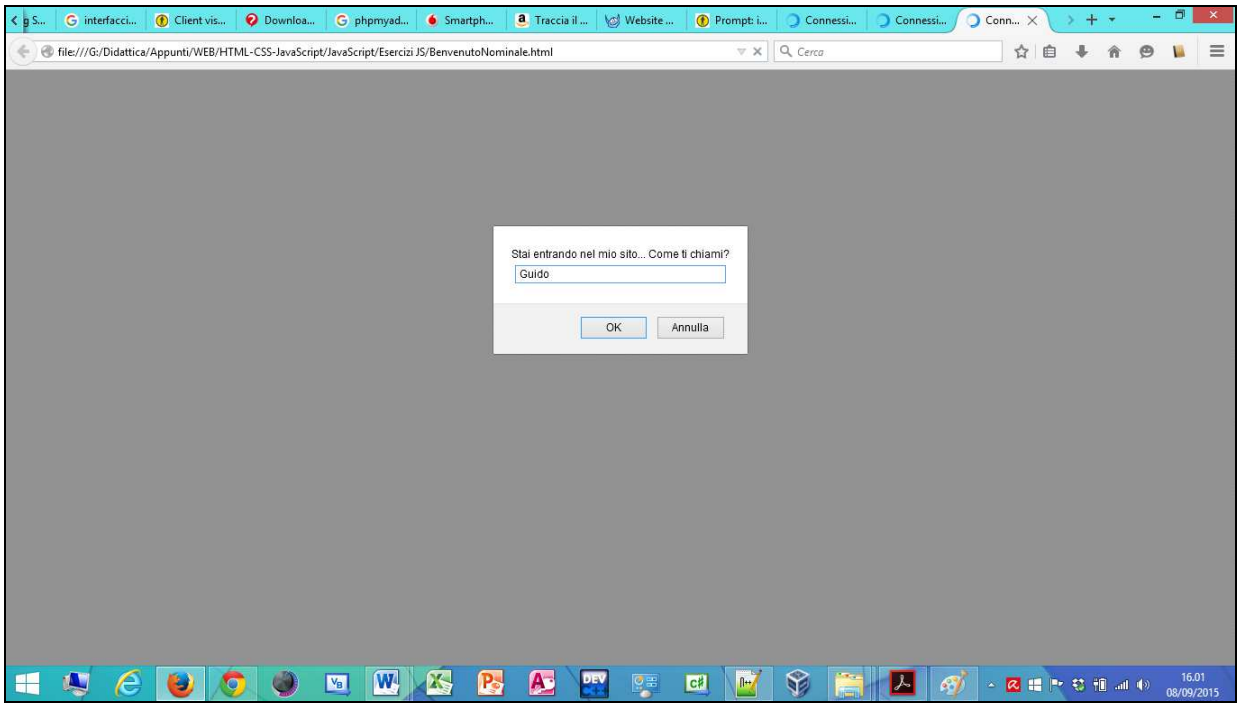
Se si vuole scrivere direttamente sul documento (la pagina html) e non più in una piccola finestra predefinita, si può usare il metodo **write()** dell'oggetto **document**.

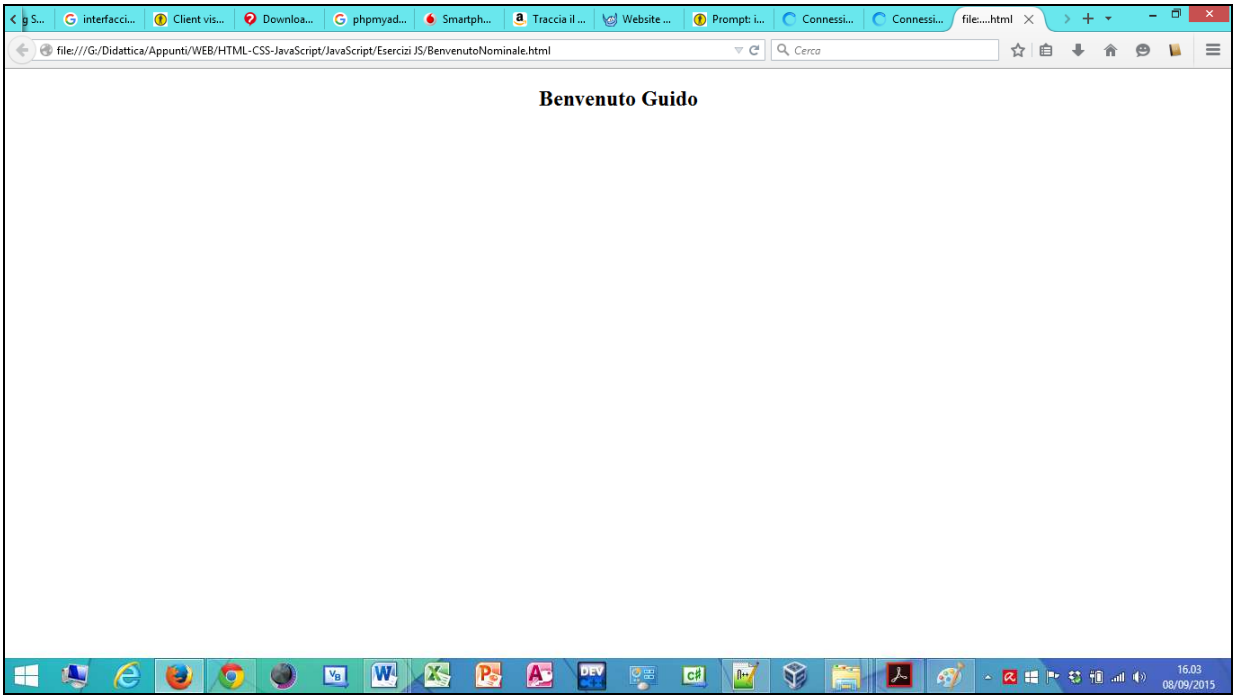
La sintassi è:

document.write (Stringa)

Esempio:

```
<!doctype HTML>

<html lang="it">

<head>
        <meta charset="UTF-8" />
        <script type="text/javascript">
                var Nome=window.prompt("Stai entrando nel mio sito... Come ti chiami?")
        </script>
</head>

<body>
        <h2 align="center">
        <script type="text/javascript">
                document.write ("Ciao "+Nome)
        </script>
        </h2>
</body>

</html>
```





Con questo metodo possiamo scrivere anche codice HTML, ad esempio per dare un comando
 (la riga javascript non va a capo)

Se il codice HTML (o la scritta) all'interno del metodo document.write() contiene le virgolette, queste devono essere precedute dal "back-slash" (\). Si tratta del cosidetto "carattere di escape"

Esempio:

```
document.write("<h1><font face=\"Verdana\" size=\"5\">")
….
….
document.write("</font></h1><br>");
```

inserisce nella pagina le formattazioni
*<h1> … </h1>
*

I motori di ricerca non leggono JavaScript e quindi è bene scrivere il codice HTML attraverso questo linguaggio soltanto nel caso in cui ci siano particolari ragioni particolari per farlo.

Esecuzione di codice javascript all'interno di un tag di link

Le istruzioni Javascript possono essere inserite ed eseguite anche all'interno di un tag di link e non serve porle tra i tag <script> e </script>. La sintassi è:

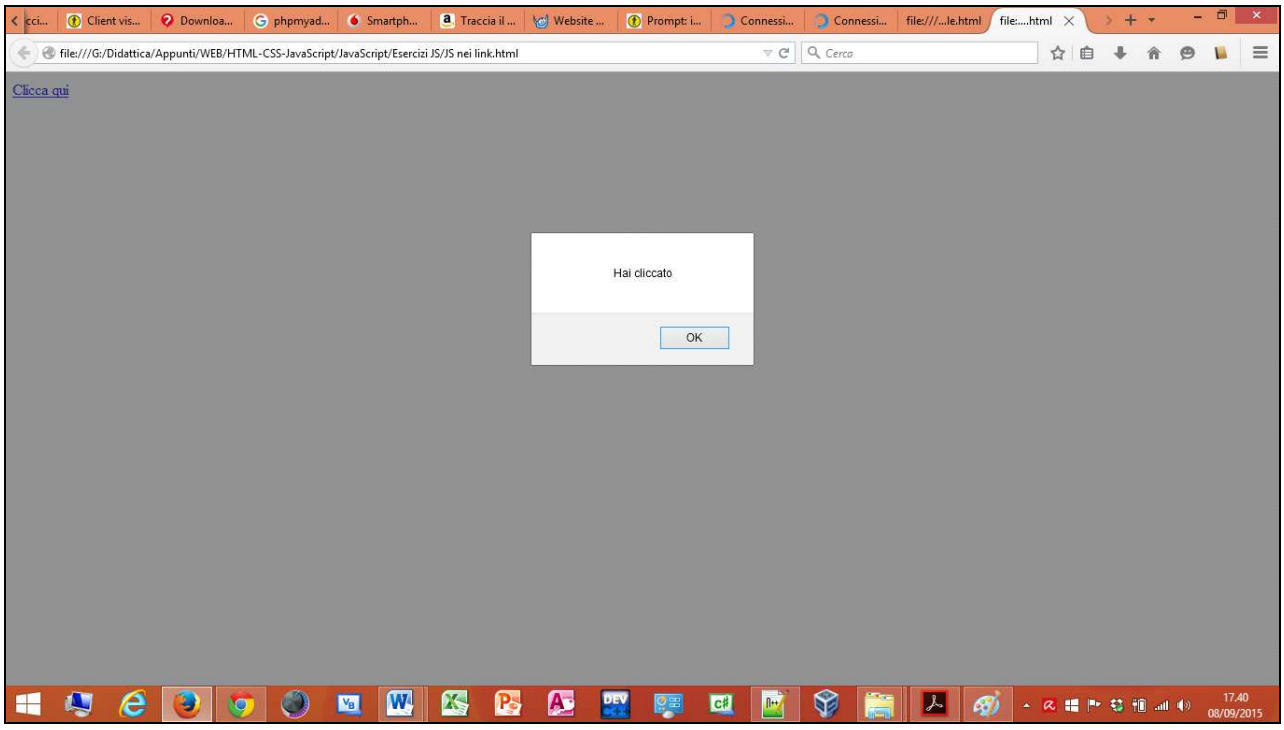
link

Le istruzioni saranno eseguite non appena si cliccherà sul link.
Nell'esempio seguente, cliccando sulla frase "clicca qui", comparirà la finestra di alert().

Esempio:

```
<!doctype HTML>
<html lang="it">
<head>
        <meta charset="UTF-8" />
</head>
<body>
    <a href= "javascript: alert('Hai cliccato')">Clicca qui</a>
 </body>
</html>
```

Se una costante stringa è inserita in un'istruzione che la raccoglie tra gli apostrofi, per la stringa si possono usare gli apici come delimitatori.



Le funzioni

Come in ogni linguaggio, anche in JavaScript è possibile creare sottoprogrammi da invocare invocare tramite una chiamata.
In Javascript funzioni e procedure utilizzano la stessa parola chiave: **function**; ad ogni modo, le procedure eseguono un blocco di codice senza restituire esplicitamente un valore, mentre le funzioni eseguono un blocco di codice restituendo uno specifico valore di ritorno.

Per dichiarare una funzione, si usa la seguente sintassi:

```
function Nomefunzione(par1,...,parN)
{
        BloccoIstruzioni
}
```

dove

- NomeFunzione è il nome assegnato al sottoprogramma;
- par1,...,parN sono gli eventuali parametri, che possono essere di qualsiasi tipo (come per le variabili, il tipo non è specificato);
- BloccoIstruzioni rappresenta il corpo della funzione, che deve essere racchiuso tra parentesi graffe.

Le funzioni sono generalmente dichiarate nella sezione head della pagina. In questo modo, esse sono lette in memoria non appena è caricata la pagina HTML, ma eseguite solo quando sono espressamente chiamate.

Se una funzione deve restituire uno specifico valore, occorre usare l'istruzione return, seguita dall'espressione che produce il valore da restituire (può essere anche una semplice variabile). La sintassi è:

return (espressione)

Esempio: cliccando su un link, si chiede all'utente di inserire due numeri, in altrettante finestre prompt. Il risultato apparirà all'interno della pagina. Le operazioni di iput e il calcolo della somma sono eseguiti attraverso la chiamata alla funzione Somma.

```
<!doctype HTML>

<html lang="it">

    <head>
      <meta charset="UTF-8" />
      <title>Esempi</title>
      <script>
            function Somma()
            {
              var x=parseInt(prompt("Inserire il primo numero",0))
              var y=parseInt(prompt("Inserire il secondo numero",0))
              var tot=x+y
              return(tot)
            }
      </script>
    </head>

    <body>
        <a href="javascript:document.write('Somma = '+Somma())">
            Clicca qui
        </a>
    </body>

</html>
```

Funzioni con parametri

Una funzione può ricevere dal programma chiamante uno o più parametri su cui operare. I parametri sono posti, separati da una virgola, all'interno delle parentesi tonde poste a fianco del nome della funzione stessa.

Esempio: cliccando su un link, si chiede all'utente di inserire due numeri, in altrettante finestre prompt. Il risultato apparirà all'interno della pagina. Questa volta il chiamante esegue le operazioni di input, mentre la somma è calcolata da una funzione che riceve i numeri inseriti come parametri.

```
<!doctype HTML>
<html lang="it">
<head>
        <meta charset="UTF-8" />
        <title>Esempi</title>
        <script>
          function Somma(x,y)
          {
                var tot=x+y
                return(tot)
          }
        </script>
</head>
<body>
        <script>
                 var n1=parseInt(prompt("Inserisci il primo numero",0))
                 var n2=parseInt(prompt("Inserisci il secondo numero",0))
                 document.write("Somma = " + Somma(n1,n2))
        </script>
</body>
</html
```

Variabili locali e globali e loro visibilità

Le variabili globali sono visibili da tutte le funzioni del documento HTML e vanno dichiarate all'inizio dello script e fuori da ogni funzione; il luogo ideale è subito dopo il tag <script> della sezione <head>.
Le variabili locali sono visibili solo all'interno della funzione in cui sono dichiarate, cioè all'interno del blocco di codice compreso fra function() e la chiusura delle parentesi graffe.

Esempio: x è una variabile globale, ma la funzione Modifica() la ridefinisce come locale, per cui la modifica apportata (x=20) non è riscontrato dalla funzione Visualizza().

```
<!doctype HTML>

<html lang="it">
<head>
        <meta charset="UTF-8" />
        <title>Esempi</title>
        <script>
                var x=10
                function Modifica()
                  {
                        var x
                        x=20
                        alert("x locale vale "+x)
                  }
                function Visualizza()
                  {
                        alert("x globale vale "+x)
                  }
```

```
        </script>
</head>

<body>
        <a href="javascript: Modifica()">Funzione Modifica</a><br>
        <a href="javascript: Visualizza()">Funzione Visualizza</a>
</body>
</html>
```

Tipi di dati

I valori che possono essere assegnati alle variabili (di cui comunque non si deve definire il tipo) possono essere:

- interi (positivi e negativi);
- decimali (numeri a virgola mobile);
- logici o booleani (valori che possono assumere soltanto due stati, vero o falso, indicati con true o false, scritti in minuscolo);
- stringhe (una qualsiasi sequenza di caratteri).

Operatori

L'operatore di assegnazione è rappresentato dal simbolo = e può essere compattato, cioè abbinato a un operatore aritmetico. In genere, quando le espressioni sono del tipo:

NomeVariabile=NomeVariabile operatore espressione
(es. x=x+y)

possono essere scritte nelle seguente forma abbreviata:

NomeVariabile operatore= espressione
(es. x+=y)

Si ha, quindi:

x+=y	**x=x+y**
x-=y	**x=x-y**
x*=y	**x=x*y**
x/=y	**x=x/y**
x%=y	**x=x%y**

Gli **operatori aritmetici** si dividono in:
- operatori unari
- operatori binari.

Gli **operatori binari** si applicano a due operandi e non ne cambiano il valore. Essi sono:

+ , - , * , / , %

Gli **operatori unari** sono gli operatori di incremento **++** e di decremento **--**.

P++ equivale a P=P+1
P-- equivale a P=P-1

Gli **operatori relazionali** sono:

>,>=,<,<=,==,!=

Gli **operatori logici** sono:

||	**(OR)**
&&	**(AND)**

^ (XOR)
! (NOT)

Gli operatori && e || non valutano l'operando a destra se non è necessario.
Le tabelle di verità per gli operatori logici sono:

x	y	x&&y	x||y	x^y	!x
false	false	false	false	false	true
false	true	false	true	true	true
true	false	false	true	true	false
true	true	true	true	false	false

Costrutti della programmazione

Costrutto di selezione

La sintassi è:

```
If (espressione da valutare)
        {istruzioni da eseguire se vero}
```

oppure

```
If (espressione da valutare)
        {istruzioni da eseguire se vero}
else
        {istruzioni da eseguire se falso}
```

Esempi:

```
if  (a==1) alert("attenzione");

if (a==1)
        {alert("a vale 1");}
else
        {alert ("valore non corretto");}
```

Costrutto di selezione multipla

In presenza di più alternative è possibile utilizzare il costrutto di selezione multipla

```
Switch (espressione da valutare)
   {
      case valore1:
       {
        istruzioni da eseguire;
        break;
       }
       case valore2:
       {
          istruzioni da eseguire;
           break;
       }
       default:
       {
       istruzioni da eseguire;
       }
   }
```

Nel caso in cui il valore dell'espressione da valutare sia uguale a uno dei valori specificati dopo la parola **case**, vengono eseguite le istruzioni corrispondenti a quel valore; se nessuno dei valori proposti è incontrato, si eseguono le operazioni che seguono la parola **default** (se specificata).
Break al termine di ogni blocco fa uscire dal comando condizionato.

Esempio:

```
<!doctype HTML>
<html lang="it">
<head>
        <meta charset="UTF-8" />
</head>
<body>
 <script>
          var a,b, operatore,risultato
          a=parseInt(prompt("Inserire il primo operando",0))
          b=parseInt(prompt("Inserire il secondo operando",1))
          operatore=prompt("Inserire l'operatore","+")
          switch(operatore)
          {
           case "+":
                 {risultato=a+b
                  document.write("la somma è "+risultato)
                  break
                 }
           case "-":
                 {risultato=a-b
                  document.write("la differenza è "+risultato)
                  break
                 }
           case "*":
                   {risultato=a*b
                    document.write("il prodotto è "+risultato)
                    break
                   }

           case "/":
           {
            if (b!=0)
                {risultato=a/b
                document.write("la divisione è "+risultato)
             }
             else
                    document.write("ERRORE")
                   break
           }
           default:
            document.write("operatore errato")
          }
 </script>
</body>
</html>
```

Costrutti iterativi

- **Il ciclo for**

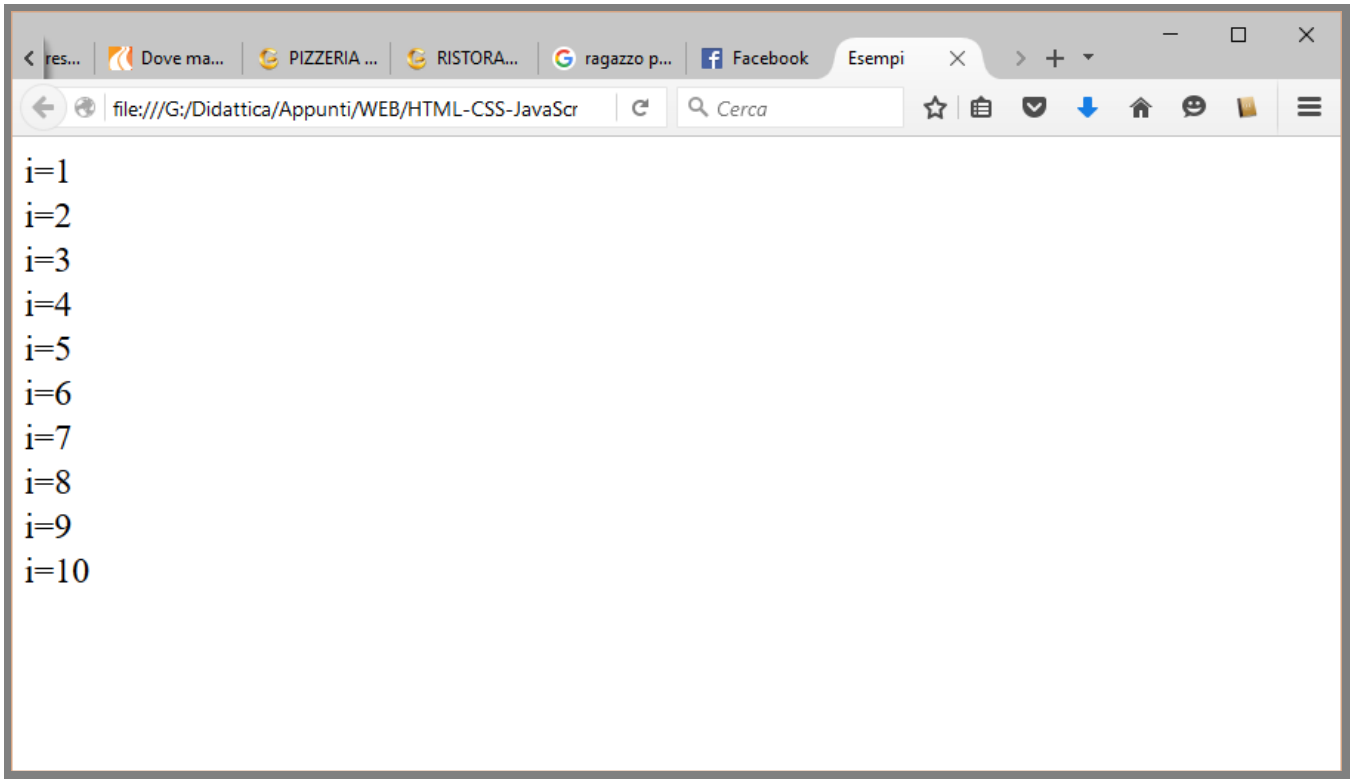
La sintassi è:

for (valore iniziale; condizione di arresto; incremento)
{istruzioni da eseguire}

Esempio: Visualizzazione dei numeri interi da 1 a 10.

```
for (var va=1; va < 11; va++)
{
        Document.write(va);
}
```

La prima assegnazione in parentesi assegna un valore iniziale alla variabile va. Il secondo termine in parentesi esprime la condizione di ripetizione del ciclo. L'ultimo termine esprime l'incremento della variabile del ciclo.



- **Il ciclo while**

È un ciclo pre-condizionato, ripete se la condizione è vera.
La sintassi è:

while (condizione)
{
istruzioni da eseguire
}

Mentre la condizione è vera, si eseguono le istruzioni racchiuse fra le parentesi graffe. Le istruzioni potrebbero non essere mai eseguite (se la condizione è subito falsa).

Esempio: Si possono ottenere I risultati dell'esempio precedente anche usando il ciclo while.

```
<!doctype HTML>
<html lang="it">
<head>
        <meta charset="UTF-8" />
</head>
<body>
```

```
        <script language="javascript">
                var i=1
                while (i<=10)
                 {
                    document.write("<font size=5>")
                    document.write("i="+i+"<br></font>")
                    i++
                 }
        </script>
</body>
</html>
```

- **Il ciclo do..while**

È un ciclo post-condizionato, ripete se la condizione è vera.
La sintassi è:

```
do
        {
                istruzioni da eseguire
        }
while  (condizione)
```

Ripete le istruzioni racchiuse fra le parentesi graffe mentre la condizione è vera. Le istruzioni sono eseguite almeno una volta.

Esempio:Vidsualizzazione dei numeri da 1 a 10 con do..while

```
<html>
<body>
    <script language="javascript">
        var va=1;
        do
            {
                document.write(va + "<br>");
                va++;
            }
        while (va<=10);
    </script>
</body>
</html>
```

In tutti i cicli, se il corpo è costituito da una sola istruzione, si possono evitare le parentesi graffe.

Stringhe

Ci sono due modi per dichiarare una variabile come stringa:

1. assegnando il valore della stringa ad una variabile.
   Esempio: test="questo è un test"

2. dichiarando un nuovo oggetto String.
   Esempio: *testo=new String ("testo inserito").*

   Oppure *testo=new String*
   testo="testo qualsiasi"

La seguente tabella riporta le proprietà e i metodi più usati per le stringhe:

Proprietà o metodo	Descrizione	Esempio
length	calcola la lunghezza di una stringa	*testo.length*
toUpperCase()	converte tutti i caratteri della stringa in lettere maiuscole	*testo.toUpperCase()* *converte ciò che è contenuto nella stringa in maiuscolo*
toLowerCase()	converte tutti i caratteri della stringa in lettere minuscole	*testo.toLowerCase()* *converte ciò che è contenuto nella stringa in minuscolo*
substring()	restituisce la porzione di stringa compresa fra le posizioni specificate dai due indici, il primo dei quali specifica la posizione di partenza e il secondo quella di arrivo (escluso il valore specificato dall'indice)	*testo.substring(3,6)* *prende i caratteri dalla posizione 3 (quarto carattere della stringa poiché la numerazione parte da 0) alla posizione 5 (perché 6 è escluso).*
charAt(n)	consente di prelevare un singolo carattere da una stringa	*testo.charAt(0)* *restituisce il primo carattere della stringa testo.*
indexOf	ricerca una sottostringa all'interno di una stringa	*loc= testo.indexOf("te")* *ricerca la stringa "te" nella stringa testo.* *testo.indexOf("te",4)* *restituisce la posizione della sottostringa "te" partendo la ricerca dalla posizione 4.*
lastIndexOf()	funziona alla stessa maniera di IndexOf() ma trova l'ultima occorrenza della stringa	*testo.lastIndexOf("te")* *testo.lastIndexOf("te",4)*

Array

E' possibile definire un array in uno dei seguenti modi:

var nomearray = new Array(num) — dove num è il numero degli elementi che costituiscono il vettore

var nomearray = new Array() — crea un array vuoto

var nomearray = new Array(val1,val2,...) — specifica un elenco di argomenti usati come valori da assegnare al vettore

L'indice parte da zero.

Esempio:

```
animals=new Array("rana","anatra","asino","orso","gallina")
```

che equivale a:

```
animals=new Array();
animals[0]="rana";
animals[1]="anatra";
animals[2]="asino";
```

```
animals[3]="orso";
animals[4]="gallina";
```

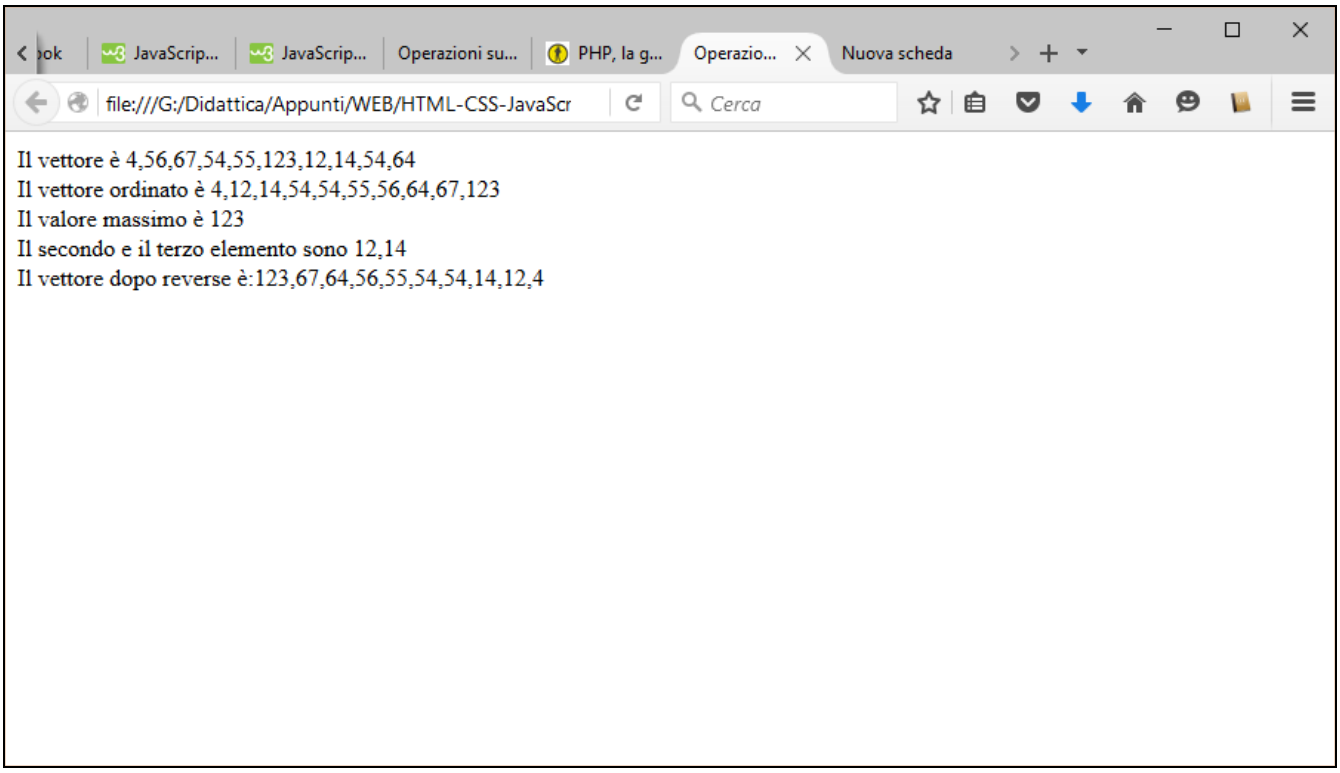
In Javascript mancano le matrici bidimensionali e tridimensionali.

La seguente tabella riporta le proprietà e i metodi più usati per gli array:

Proprietà o metodo	Descrizione	Esempio
length	Proprietà che indica il numero degli elementi dell'array.	*vettore.length*
sort()	Ordina l'array in ordine alfabetico. Per eseguire l'ordinamento numerico o per imporre un ordinamento diverso da quello crescente di default, si deve passare al metodo una funzione che definisce un ordinamento alternativo. La funzione deve restituire zero o un valore positivo o un valore negativo, ad esempio: function (a, b) { return a- b } Quando il metodo sort () confronta due valori, invia i valori alla funzione di confronto e li ordina in base al valore restituito. Ad esempio, quando si confrontano 40 e 100, il metodo sort () chiama la funzione di confronto passando i parametri (40,100). La funzione calcola 40-100 e restituisce -60 (un valore negativo). La funzione di ordinamento ordinerà allora 40 come un valore inferiore a 100. Usare return (b-a) per ordinamento decrescente.	*vettore.sort* *o* *vettore.sort(NomeFunzione)* *dove NomeFunzione può essere definita così:* *function NomeFunzione (a,b) {return b-a}* *O* *Vettore.sort (function (a,b){return a-b})*
join()	Converte gli elementi di un array in una stringa concatenata. Per separare gli elementi, per default è utilizzata la virgola, ma è comunque possibile specificare un carattere (o stringa) differente.	*document.write("L'elenco degli elementi del vettore è:"+vettore.join("-")+" ")*
reverse()	Inverte l'ordine degli elementi nell'array	*document.write("Il vettore dopo reverse è:" + vettore.reverse()+" ")*
slice()	Restituisce, in un nuovo array, un sottoinsieme di elementi dall'array specificato; al metodo si passano due argomenti: indice di inizio e indice di fine (non incluso), determinando così i valori da estrapolare. È anche possibile passare al metodo un solo parametro, in questo caso il sottoinsieme estrapolato sarà quello dalla posizione indicata alla fine dell'array.	*vett2=vettore.slice(0,3)*
splice()	Rimuove elementi dall'array specificato. Restituisce l'array degli elementi eliminati o l'array completamente vuoto, se non è stato eliminato alcun elemento. Il primo argomento specifica la posizione dell'array da cui partire; il secondo specifica il numero di elementi da eliminare; se questo viene omesso saranno eliminati tutti gli elementi a partire dall'elemento iniziale fino alla fine dell'array stesso.	*Vettore2=vettore.splice(0,2)*

Esempio: *Operazioni sui vettori e uso del metodo sort per ordinare in senso numerico.*

```
<!doctype HTML>

<html lang="it">

    <head>
        <title>Operazioni sui vettori</title>
        <meta charset="UTF-8" />
    </head>

    <body>
        <script language="javascript">
            var vettore=new Array()
            var i=1
            while (i<=10)
            {
                vettore[i-1]=parseInt(prompt("inserisci il "+i+"° valore"))
                i++
            }
            document.write("Il vettore è " + vettore + "<br>")
            document.write("Il vettore ordinato è " + vettore.sort((function (a,b){return a-b})) + "<br>")
            document.write("Il valore massimo è " + vettore[9] + "<br>")
            document.write ("Il secondo e il terzo elemento sono " + vettore.slice(1,3) +"<br>")
            document.write("Il vettore dopo reverse è:" + vettore.reverse() + "<br>")
        </script>
    </body>
```



Diversamente dagli altri linguaggi di programmazione, gli array Javascript:

- hanno una dimensione dinamica:
- l'array può contenere anche elementi di tipo diverso.
  Es. var A= new Array(1, "pippo", true)

Oggetti del contenitore FORM

Tramite la proprietà name ci si riferisce agli oggetti del form.
Scrivendo

f1.txtnome.value

si fa riferimento al contenuto della casella di testo di nome *txtnome* presente nel form *f1*

- **L'oggetto button**

L'oggetto button permette di gestire tutti i pulsanti di un form HTML, anche i pulsanti speciali SUBMIT e RESET.
Le sue proprietà sono: **name**, **type** e **value**, riflesse dagli attributi NAME, TYPE e VALUE del rispettivo elemento HTML.

- **L'oggetto casella di testo**

Proprietà: **name**, **type**, **value** e **size** che riflettono gli attributi NAME, TYPE, VALUE e SIZE del rispettivo elemento HTML.

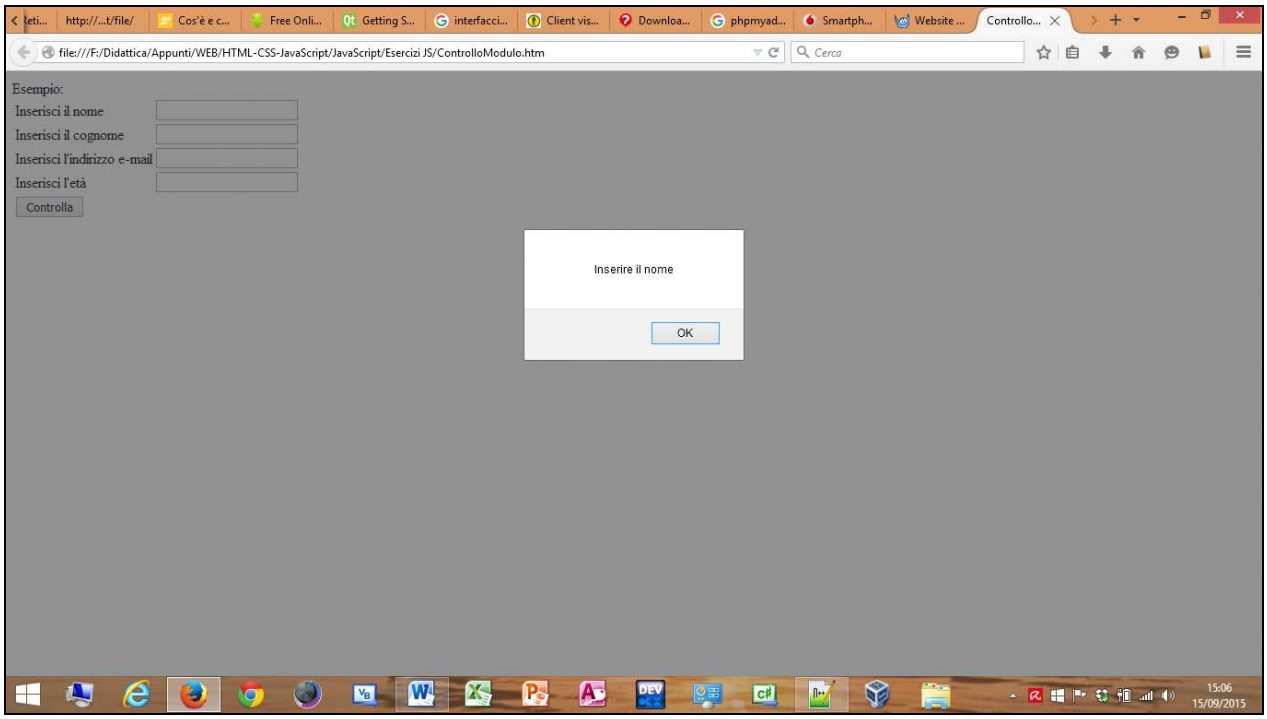
Esempio: controllo dei dati inseriti in un form.

```
<!doctype HTML>
<html lang="it">
<head>
        <title>Controllo form</title>
        <meta charset="UTF-8" />
        <script>
                function Controlla()
                {
                  if (f1.txtnome.value=="")
                        alert("Inserire il nome")
                  else
                        if (f1.txtcognome.value=="")
                              alert("Inserire il cognome")
                        else
                          if (f1.txtmail.value.indexOf("@",0)==-1)
                                alert("Indirizzo e-mail non corretto ")
                          else
                                if (f1.txteta.value=="")
                                      alert("Inserire l'età")
                                else
                                  if ((isNaN(f1.txteta.value)) || (parseInt(f1.txteta.value)<0) || (parseInt(f1.txteta.value)>120))
                                        alert ("Inserire un valore corretto per l'età")
                                  else
                                        {alert("OK dati corretti")
                                        document.forms["f1"].submit();}
                }
        </script>
</head>
<body>
        <form name="f1" method="post" action="x.php">
                <table>
                        <tr>
                                <td>Inserisci il nome</td>
                                <td><input type="text" name=txtnome></td>
                        </tr>
                        <tr>
                                <td>Inserisci il cognome</td>
                                <td><input type="text" name=txtcognome></td>
```

isNaN(n) è una funzione booleana che determina se n è un numero corretto o no (Not-a-Number). Restituisce true se il numero non è valido.

**document.forms["f1"].submit()
fa in modo che il pulsante di tipo "button" si comporti come "submit", se tutti i dati sono stati inseriti in modo corretto.**

```
                </tr>
                <tr>
                        <td>Inserisci l'indirizzo e-mail</td>
                        <td><input type="text" name=txtmail></td>
                </tr>
                <tr>
                        <td>Inserisci l'et&agrave;</td>
                        <td><input type="text" name=txteta></td>
                </tr>
                <tr>
                        <td colspan="2">
                           <input type="button" name="test" value="Controlla" onclick="Controlla()">
                        </td>
                </tr>
        </table>
    </form>
</body>
```



- **L'oggetto area di testo**

Proprietà: **name**, **value**, **size**, **cols**, **rows** riflesse dagli attributi NAME, TYPE, VALUE, SIZE, COLS e ROWS del rispettivo elemento del codice HTML.

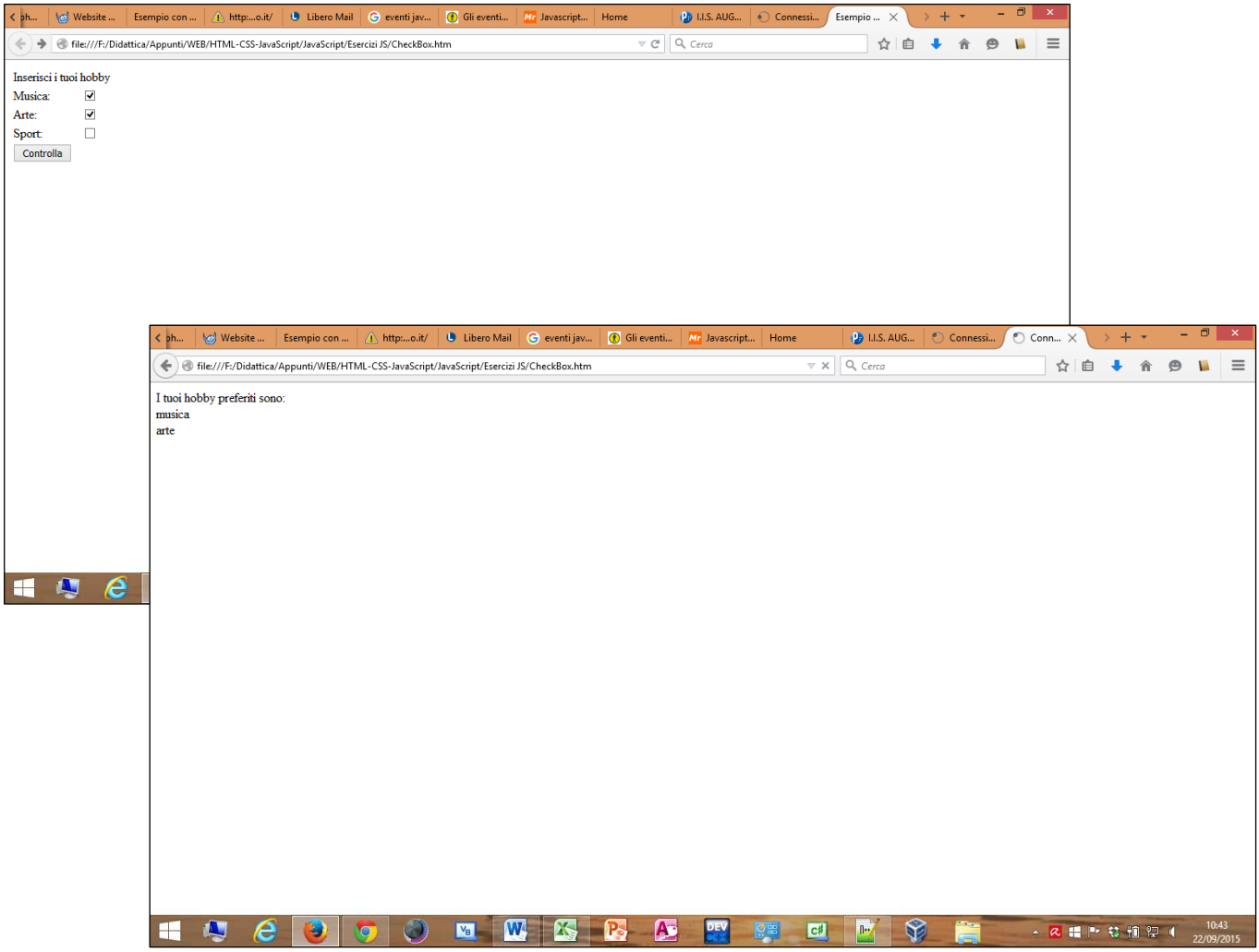
- **L'oggetto caselle di controllo**

Proprietà: **checked**, valore booleano true o false a seconda che il pulsante di opzione sia selezionato o meno, **name**, **type**, **value**, riflesse dagli attributi NAME, TYPE, VALUE del rispettivo elemento HTML.
Se più caselle di controllo hanno lo stesso nome, possono essere individuate attraverso un indice, come elementi di un array.

Esempio:

```
<!doctype HTML>
```

```
<html lang="it">

<head>
	<meta charset="UTF-8" />
	<title>Esempio con checkbox</title>
	<script>
		function Controlla()
		{
		    var hob="", k=0
		    for (var i=0; i<3; i++)
			    if (f1.chkhobby[i].checked)
			        {
			           k=k+1
			           hob=hob+f1.chkhobby[i].value+"<br> "
			        }
		    if (k==1)
		       document.write ("Il tuo hobby preferito e': <br>" +hob)
		    else
		       if (k>1)
		         document.write ("I tuoi hobby preferiti sono: <br>" +hob)
		       else
		         document.write ("Ma come, non hai un hobby?")
		}
	</script>
</head>

<body>
	<form name="f1">
	  <table>
		<tr>
			<td colspan="2">Inserisci i tuoi hobby</td>
		</tr>
			<tr>
			<td>Musica:</td>
			<td><input type="checkbox" name="chkhobby" value="musica"></td>
		</tr>
		<tr>
			<td>Arte:</td>
			<td><input type="checkbox" name="chkhobby" value="arte"></td>
		</tr>
		<tr>
			<td>Sport:</td>
			<td><input type="checkbox" name="chkhobby" value="sport"></td>
		</tr>
		<tr>
			<td colspan="2">
			   <input type="button" name="btnControlla" value="Controlla" onclick="Controlla()">
			</td>
		</tr>
	  </table>
	</form>
</body>
```



- **L'oggetto pulsante di opzione**

Gli oggetti radio button che fanno parte dello stesso gruppo (cioè mutuamente esclusivi) devono avere lo stesso nome.
Proprietà: **checked**, valore booleano true o false, a seconda che il pulsante di opzione sia selezionato o meno, **name**, **type**, **value**, riflesse dagli attributi NAME, TYPE, VALUE e SIZE del rispettivo elemento HTML; **length** rappresenta il numero di pulsanti di opzione presenti in un oggetto radio.

- **L'oggetto lista di selezione**

L'oggetto **select** è un contenitore di oggetti **option**, ai quali si può accedere usando l'array **options**.
Se un oggetto select, chiamato "sel", contiene due opzioni, queste vengono riflesse in Javascript con

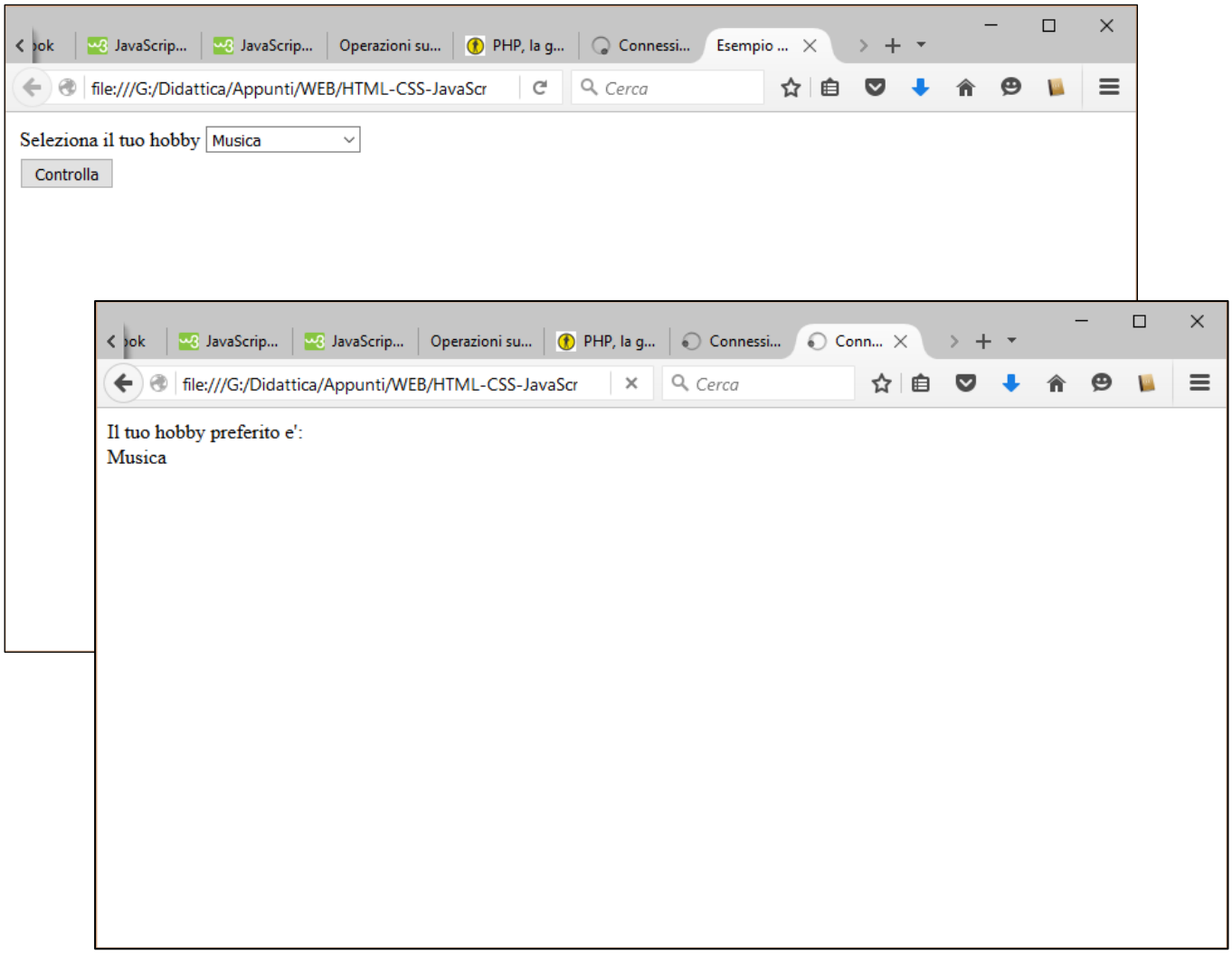
sel.options[0] e sel.options[1].

Proprietà:

- **name** e **value**, riflesse dagli attributi NAME, VALUE del rispettivo elemento HTML;
- **length**: contiene il numero di OPTION contenuto nell'elemento SELECT dell'HTML;
- **options**: riflette il valore di ogni elemento OPTION, cioè il testo specificato dopo il tag di apertura <OPTION>;
- **selectedindex**: contiene l'indice dell'opzione selezionata;
- **text**: riflette il testo che segue il tag <option>

Esempi:

1. Si seleziona dalla ComboBox l'hobby d'interesse e, tramite una funzione (Controlla), è visualizzato a video:




```
<!doctype HTML>
<html lang="it">
<head>
    <title>Esempio con select</title>
    <meta charset="UTF-8" />
    <script>
        function Controlla()
            {
                var hob=f1.cmbhobby.options[f1.cmbhobby.selectedIndex].value
                if (hob=="Non ho un hobby")
                    document.write ("Ma come, non hai un hobby?")
                else
                    document.write ("Il tuo hobby preferito e': <br>" +hob)
            }
    </script>
</head>
<body>
<form name="f1">
  <table>
    <tr>
```

selectedIndex è la proprietà della ComboBox che individua la posizione dell'opzione selezionata

```
        <td colspan="2">Seleziona il tuo hobby</td>
          <td>
                <select name="cmbhobby">
                  <option value="Musica" selected>Musica</option>
                  <option value="Sport">Sport</option>
                  <option value="Arte">Arte</option>
                  <option value="Lettura">Lettura</option>
                  <option value="Cinema">Cinema</option>
                  <option value="Teatro">Teatro</option>
                  <option value="Non ho un hobby">Non ho un hobby</option>
                </select>
          </td>
        </tr>
        <tr>
          <td colspan="2">
           <input type="button" name="test" value="Controlla" onclick="Controlla()">
          </td>
        </tr>
      </table>
    </form>
    </body>
    </html>
```

2. Si seleziona dalla ComboBox l'hobby d'interesse e la funzione Controlla esegue un link alla pagina corrispondente, diversa a seconda della scelta effettuata. Se l'utente seleziona "Non ho un hobby", compare un messaggio.

```
<!doctype HTML>
<html lang="it">
<head>
    <title>Esempio con select</title>
    <meta charset="UTF-8" />
    <script>
          function Cambiapagina()
                {
                 var hob=f1.cmbhobby.options[f1.cmbhobby.selectedIndex].value
                 if (hob == "Nessuno")
                       alert("Peccato, non hai un hobby!")
                 else
                       window.location=hob
                }
    </script>
</head>
<body>
<form name="f1">
<table>
    <tr>
          <td colspan="2">Seleziona i tuoi hobby</td>
          <td>
                <select name="cmbhobby">
                  <option value="musica.htm">Musica</option>
                  <option value="sport.htm">Sport</option>
                  <option value="arte.htm">Arte</option>
                  <option value="lettura.htm">Lettura</option>
                  <option value="Cinema.htm">Cinema</option>
                  <option value="Teatro.htm">Teatro</option>
                  <option value="Nessuno">Non ho un hobby</option>
                </select>
          </td>
```

window.location reindirizza il browser a una nuova pagina di cui si specifica l'url.

```
    </tr>
    <tr>
      <td colspan="2">
            <input type="button" name="test" value="Controlla" onclick="Cambiapagina()">
      </td>
    </tr>
  </table>
  </form>
  </body>
```

Eventi

Come visto, le istruzioni in Javascript possono essere eseguite in seguito all'attivazione di un evento, come un click del mouse (gestore evento **onclick**) o la pressione di un tasto.

Javascript supporta moltissimi eventi che, con l'evolversi del linguaggio, sono andati via via arricchendosi.

La sintassi degli eventi Javascript

L'evento si richiama direttamente all'interno dell'elemento HTML che lo riguarda, incorporando un attributo nel codice dell'elemento stesso; la sintassi è la seguente:

<elemento attributoEvento="istruzione Javascript">

Esempi:

```
<a HREF="pagina.html" onclick="alert('ciao')">Link</a>
```
Cliccando sul link compare una finestra di alert con scritto 'Ciao'.

```
<a HREF="link.htm" onclick="return(confirm('Sei sicuro?'))"> Link</a>
```
Se si risponde OK il link si attiva, se si risponde Annulla il link non si attiva.

```
<input type="button" value="Invia" onclick="alert('Hai cliccato!')">
```
Cliccando sul pulsante Invia, compare una finestra di alert con scritto 'Ciao'.

Dato che l'HTML non è un linguaggio case-sensitive (sensibile alle maiuscole ed alle minuscole) è indifferente scrivere onClick, onclick oppure ONCLICK, ma è necessario specificare i parametri (ad esempio il nome della funzione) con le giuste maiuscole e minuscole all'interno di un evento (Javascript è case-sensitive).

I principali eventi Javascript

Si riporta di seguito un elenco degli eventi e dei relativi gestori, con opportuna descrizione.

- **Eventi del tag <body>: onload, onunload, onresize e onscroll**

Due eventi molto utilizzati in associazione al corpo della pagina sono **onload** e **onunload** attraverso i quali si intercetta, rispettivamente, l'apertura e la chiusura della pagina.
Esempi:

```
<body onload="alert('Benvenuto!');">
```
Messaggio di saluto all'apertura della pagina.

```
<body onunload="alert('Torna a trovarci!');">
```
Messaggio di saluto alla chiusura della pagina.

Da segnalare che l'evento *onload* può essere efficacemente applicato anche al tag <img> per identificare il momento in cui l'immagine viene effettivamente scaricata dal browser.

Gli altri eventi associabili al corpo della pagina sono **onresize** e **onscroll**, che servono, rispettivamente, ad intercettare il ridimensionamento della finestra e lo scrolling al suo interno.

- **Gli eventi legati al mouse: onclick, onmouseover, onmouseout, ecc.**

L'attività del mouse è, solitamente, uno dei fattori più rilevanti all'interno di buona parte delle applicazioni web. Un click su un elemento o il passaggio del cursore in una determinata porzione della pagina, infatti, sono tra gli eventi più comunemente utilizzati. I più importanti di questi sono:

onclick - click col tasto sinistro del mouse;
ondblclick - doppio click col tasto sinistro del mouse;
oncontextmenu - click col tasto destro del mouse;
onmouseover - ingresso del mouse su un dato elemento;
onmouseout - uscita del mouse da un dato elemento;
onmousemove - il cursore del mouse si sta muovendo sopra un dato elemento;

Questi eventi possono essere associati ad una molteplicità di tag HTML come, ad esempio, immagini, div, paragrafi, bottoni, ecc.

Esempi:
<img src="foto.jpg" onclick="alert('Hai cliccato sulla foto!')">
<img src="foto.jpg" ondblclick="alert('Hai fatto doppio click sulla foto!')">
<img src="foto.jpg" onmouseover="alert('Sei passato col mouse sulla foto!')">
<img src="foto.jpg" onmouseout="alert('Hai spostato il mouse fuori dalla foto!')">

- **Eventi legati alla tastiera: onkeypress, onkeydown e onkeyup**

Un'altra serie importante di eventi è generata dalla pressione, da parte degli utenti, dei tasti della tastiera del computer. Gli eventi rilevanti sono tre:

onkeypress - generica pressione di un tasto;
onkeydown - l'evento si genera mentre l'utente sta premendo un tasto;
onkeyup - l'evento si scatena quando l'utente rilascia un tasto che ha premuto;

Esempio:
<input type="text" onkeydown="alert('hai premuto un tasto!')">

- **Eventi legati ai form: onsubmit, onreset, onchange, ecc.**

Un'altra fitta serie di eventi Javascript è quella che riguarda i form ed i loro elementi.
I due eventi che riguardano il tag <form> sono **onsubmit** e **onreset**, utilizzati, rispettivamente, per intercettare l'invio ed il reset di un modulo HTML.

Esempi:
<form onsubmit="alert('Hai inviato il modulo!')">
<form onreset="alert('Hai cancellato il modulo!')">

Gli eventi che riguardano i singoli elementi dei form (input, textarea, select, ecc.) sonomolteplici. I più utilizzati sono:

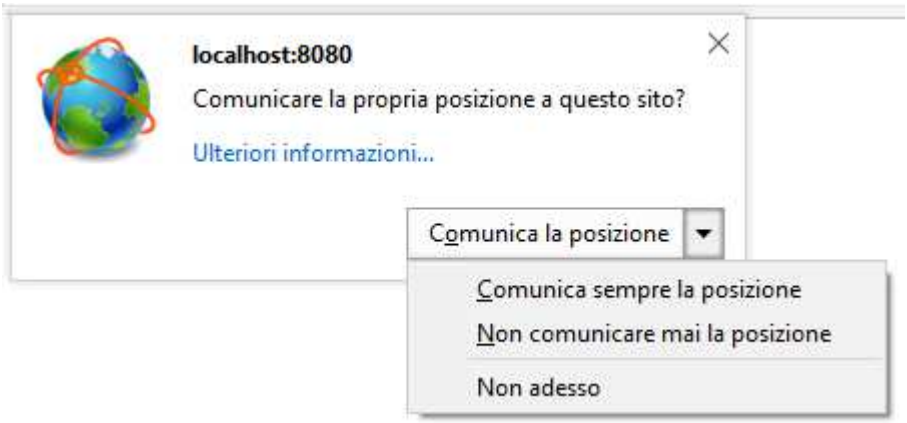
onfocus - l'elemento viene selezionato (e quindi attivato);
onblur - l'elemento perde il focus (e quindi disattivato);
onchange - l'elemento viene modificato nel contenuto;

Ad esempio, un campo *input* acquista il *focus* quando ci si clicca sopra per iniziare a scriverci dentro, lo perde quando si clicca su un altro elemento.
Per quanto riguarda l'evento *change*, questo è tipico delle *selectbox*, dei campi *input* e delle *textarea* e si verifica quando, a seguito dell'azione dell'utente, si ha un cambiamento del loro contenuto.

Geolocalizzazione con HTML5 e Javascript e integrazione con Google Maps

Tra le novità in via d'introduzione con **HTML5** spicca, senza dubbio, il supporto per la **geolocalizzazione** la quale consente di individuare automaticamente la posizione geografica dell'utente attraverso il browser.

Mediante la geolocalizzazione, infatti, i browser con supporto per HTML5 sono in grado di risalire alla posizione dell'utente mediante l'indirizzo IP assegnato dal provider di connessione oppure tramite l'antenna GPS eventualmente integrata nel dispositivo. Il rilevamento della posizione geografica avviene solo a seguito di esplicito assenso del diretto interessato, che sarà avvisato della richiesta della pagina web di effettuare il tracciamento della posizione, richiesta alla quale potra acconsentire oppure no.



La posizione geografica restituita dal browser è espressa sotto forma di coordinate (latitudine e longitudine) che potranno poi essere utilizzate mediante **Javascript**. Le nuove API per la geolocalizzazione si basano su una proprietà dell'oggetto ***navigator***:

navigator.geolocation.

Al fine di verificare se il browser offre supporto a queste nuove API di HTML5 è sufficiente, quindi, effettuare una chiamata condizionale di questo tipo:

```
if (navigator.geolocation) {
        // browser HTML5-ready con supporto per la geolocalizzazione
        …
}else{
        // nessun supporto alla geolocalizzazione
        …
}
```

Qualora il browser supporti la geolocalizzazione, si richiama la funzione

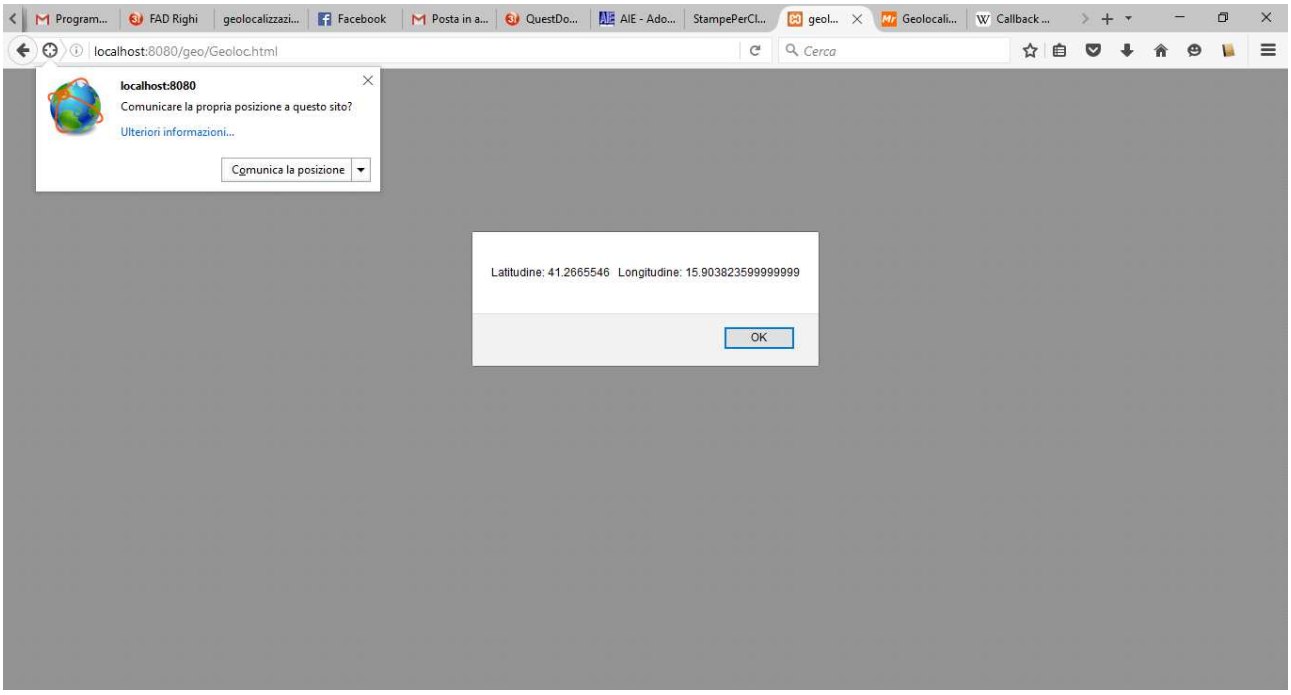
getCurrentPosition()

alla quale viene trasmesso, quale argomento, il callback in caso di successo, vale a dire il nome di una funzione da eseguire se la localizzazione ha avuto esito positivo. Questa funzione avrà un parametro che conterrà le informazioni restituite dalla procedura di geolocalizzazione.

Con un semplice script Javascript (da inserire nell'header della pagina) si può allora eseguire il rilevamento delle coordinate geografiche dell'utente:

```
<!doctype html>
<html lang="it">
<head>

        <meta charset="utf-8">
 <title> geolocalizzazione</title>

 <script type="text/javascript">
        // var myTimer = setTimeout(Localizzami, 5000);

        if (navigator.geolocation)
        {
          navigator.geolocation.getCurrentPosition(Localizzami);
        }
        Else
        {
          alert(`Geolocalizzazione non possibile');
        }

        function Localizzami(posizione) {
          var lat = posizione.coords.latitude;
          var lon = posizione.coords.longitude;
          alert ('La tua posizione: ' + lat + ',' + lon);
        }
       </script>

</head>
<body>
</body>

</html>
```

La funzione callback si chiama, nell'esempio, Localizzami, il parametro si chiama posizione e i valori che restituisce sono latitudine e longitudine, assegnate alle variabili lat e lon, delle quali viene visualizzato il valore.



Il passaggio successivo potrebbe essere la visualizzazione della posizione sulla google map.

A questo scopo occorre:

- Predisporre un oggetto sulla pagina per la visualizzazione, ad esempio un iframe, che si può impostare nel corpo della pagina, con un id che lo identifichi per l'impostazione degli stili e del contenuto:

```
<body>
        <iframe id="mappa"/>
</body>
```

- Nell'intestazione della pagina, si può aggiungere lo stile dell'iframe, impostandone in particolare le dimensioni:

```
<style>
    #mappa{width:500px; height:500px;}
</style>
```

- Nello script JavaScript, dopo aver visualizzato le dimensioni, si può visualizzare nell'iframe la mappa con la posizione individuata, recuperando da codice l'oggetto iframe attraverso getElementById e assegnando alla sua proprietà src, come valore, la pagina Google che mostra la mappa (**https://maps.google.it/maps**), con la specifica del parametro q, cui si assegna la coppia delle coordinate (latitudine, longitudine), separate da una virgola. Ancora di seguito si possono specificare lo zoom di visualizzazione (**z=17**) e il tipo di output (**output=embed**)

```
document.getElementById("mappa").src=
    'https://maps.google.it/maps?q='+lat+','+lon+'&z=17&output=embed'
```

Il codice completo diventa allora:

```
<!doctype html>
<html lang="it">
<head>

    <meta charset="utf-8">

    <title> geolocalizzazione</title>

     <style>
        #mappa{width:500px; height:500px;}
     </style>
     <script type="text/javascript">
     if (navigator.geolocation)
     {
       navigator.geolocation.getCurrentPosition(Localizzami);
     }
     Else
     {
       alert(`Geolocalizzazione non possibile');
     }

     function Localizzami(posizione) {
       var lat = posizione.coords.latitude;
       var lon = posizione.coords.longitude;
       //alert ('La tua posizione: ' + lat + ',' + lon);
            document.getElementById("mappa").src=
              'https://maps.google.it/maps?q='+lat+','+lon+'&z=17&output=embed
              '
```

```
        }
      </script>

</head>
<body>
        <iframe id="mappa"/>
  </body>

</html>
```