Tecnologie del WEB

HTML, CSS, JavaScript e PHP per esempi

2014.10

Indice

Un riassunto generale

In questi appunti viene presentata una esposizione delle tecnologie che stanno alla base del Web e che ne rappresentano, anche, l'evoluzione dagli inizi ad oggi. Per una maggiore comprensione è necessario avere nozioni di programmazione e di SQL.

Il Web nasce dall'esigenza di *condivisione* di documenti e dalla diffusione di Internet che permetteva di realizzare in maniera estesa tale obiettivo. L'obiettivo della condivisione ha portato alla esigenza di progetto di nuove tecnologie per la stesura e la conservazione dei documenti: non si possono utilizzare tecnologie proprietarie che porterebbero all'esclusione aprioristica di coloro che non li utilizzano per motivi qualsiasi. È necessario utilizzare formati accessibili da qualsiasi piattaforma hardware e software: i sistemi connessi in rete sono molto diversi fra di loro e se si vogliono raggiungere tutti si deve effettuare una specie di *raccoglimento a fattore comune*, utilizzare tecnologie disponibili a tutti liberamente. Il fattore comune dei sistemi è il codice ASCII, la codifica universalmente utilizzata, si potrebbe dire, dalla quasi totalità dei sistemi a prescindere dalla grandezza o dal costruttore. Il documento dovrà dunque essere redatto utilizzando soltanto caratteri ASCII. Gli attributi particolari di visualizzazione dei documenti (colori del testo, grassetto, sottolineato ...), che dipendono fortemente dalle caratteristiche hardware e dalla gestione del software specifico della grafica, vengono demandati al sistema che deve visualizzare il documento, che conosce il proprio hardware e che è in grado di gestirlo. Nascono le pagine HTML che sono composte da *testo ASCII puro* (senza formattazione) contenente le indicazioni (*tag*) per il software che si dovrà occupare della visualizzazione. Un apposito programma (il browser) interpreta i tag e presenta la pagina.

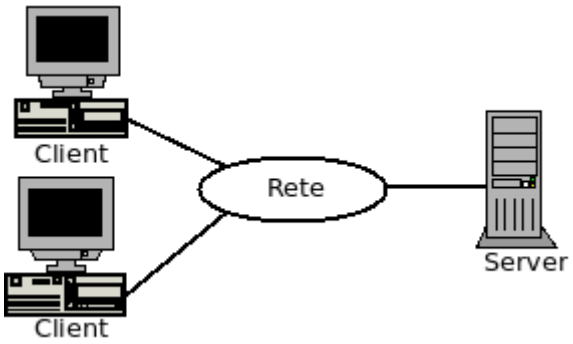
Internet si espande in maniera inimmaginabile e la possibilità di raggiungere utenti in qualsiasi parte del mondo utilizzando tecnologia a bassissimo costo aumenta a dismisura gli utenti che la utilizzano e le aspettative sulla rete. La rete originariamente nasce in ambiti di ricerca e viene resa disponibile gratuitamente, abbastanza velocemente nascono anche aziende private fornitrici di connettività per coloro che non fanno parte di Centri di Ricerca o Università, gli ISP o Internet Service Provider. Le aziende commerciali vedono le grandi potenzialità, aumentano la propria presenza sulla rete ma hanno esigenze di presentazione delle pagine, che pubblicizzano i propri prodotti, molto più sofisticate di quanto non ne abbiano i documenti tecnici prodotti da ricercatori. Il linguaggio HTML, di conseguenza a queste spinte, espande le sue potenzialità cercando di coprire le esigenza che via via si manifestano: i tag cominciano a cambiare il loro ruolo e diventano sempre più affollati di opzioni di formattazione sofisticata facendo sorgere anche problemi di compatibilità fra opzioni dei tag stessi e il modo di visualizzarli nei diversi software che hanno questo compito (i browser). Tutto ciò però complica la produzione delle pagine e va in contraddizione con l'aumento del bacino di utenza che richiede la possibilità di produrre facilmente pagine di alta qualità. L'introduzione di nuovi tag o di nuove opzioni porta una complicazione notevole per gli sviluppatori di siti web che sono costretti a ripetere i tag ogni volta: lo sviluppo diventa un vero e proprio incubo. Per risolvere il problema il consorzio W3C, coordinato e diretto da Tim Berners Lee l'inventore del WEB, sviluppa una nuova tecnologia che affianca HTML: i CSS. In pratica si tratta di distinguere il contenuto della pagina (di cui si occupa HTML) dall'aspetto grafico (compito demandato ai CSS) e in questo modo una nuova figura professionale, il Web Designer, può progettare l'aspetto grafico della pagina mentre altri si possono occupare esclusivamente del riempimento della pagina di contenuti.

Il vantaggio reale conseguente al portare, per mezzo della rete, la propria presenza nella scrivania

del cliente in qualsiasi parte del mondo si trovi, può essere assolto completamente se si ha la possibilità di interazione fra cliente e azienda. HTML permette, dal punto di vista dell'azienda commerciale, soltanto un ambiente *espositivo*. È un po' come un catalogo: si può esporre la merce ma il resto dei contatti commerciali devono avvenire per mezzo dei canali tradizionali e ciò riduce i vantaggi che si possono trarre dalla rete. Ad HTML si affiancano altre tecnologie che permettono una interazione con le pagine: il cliente può selezionare la visualizzazione selettiva della merce che interessa, può effettuare l'ordine. Vengono sviluppati, fra l'altro, JavaScript, che permette di far diventare una pagina web un programma che interagisce con l'utilizzatore, e PHP, per l'interazione con il Database dei prodotti residente nel server dell'azienda.

Modello Client/Server e WEB

La necessità di collegare vari computer fra di loro e creare una rete è la risposta a due tipi di esigenze: la comunicazione e la condivisione di risorse. Il modello più comune, nelle reti in generale e in Internet in particolare, è il modello Client/Server.

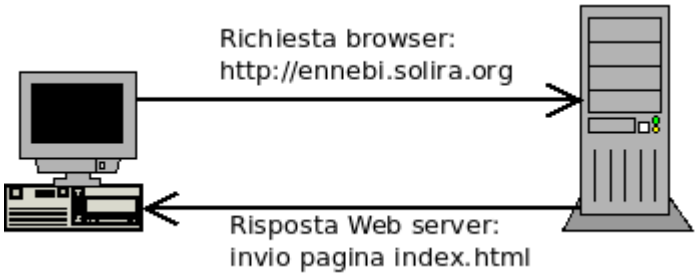


Nel modello C/S in una macchina, generalmente abbastanza potente da poter fornire servizi più velocemente possibile e in grado di soddisfare più richieste possibili, gira uno o più programmi server che mettono a disposizione servizi a chi ne fa richiesta: potrebbe, per esempio, esserci un server di stampa che fornisce la possibilità di stampare, sulla stampante collegata alla macchina, ai programmi, residenti su altre macchine, che ne facciano richiesta. Altri programmi (client) girano su macchine anche più semplici e richiedono al server i servizi disponibili.

Nel modello C/S dialogano due processi: il processo client manda una richiesta al processo server e attende una risposta, il processo server attende *in ascolto* delle richieste da parte dei client. Quando la richiesta arriva fornisce la risposta o il servizio.

Uno dei servizi fra quelli più utilizzati universalmente e presente nelle rete Internet è il World Wide Web. Progettato nel 1990 presso il CERN di Ginevra da Tim Berners Lee per permettere il reperimento e la consultazione di pubblicazioni tecniche e scientifiche (condivisione del sapere). Si tratta di legare il concetto di ipertesto alla rete facendone, in pratica, un unico ipertesto globale. Le pagine web sono composte da testo in formato ASCII puro e contengono dei marcatori (tag) che specificano l'aspetto estetico che dovrà avere il testo contenuto nella pagina visualizzata (HTML). Generalmente, e tranne pochi casi, i tag sono specificati in coppia in modo da delimitare il testo cui vanno applicati. I tag sono racchiusi fra parentesi angolari (< e >) e il tag di chiusura racchiude fra parentesi angolari lo stesso tag di apertura preceduto dal carattere di chiusura (/). Per esempio la

coppia di tag `<body>` e `</body>` racchiude la parte della pagina web che dovrà essere visualizzata. Le pagine sono gestite da un server (web server) che ne invia una copia, dietro richiesta, ad un client.



Il browser e il Web Server comunicano utilizzando il protocollo HTTP (Hyper Text Transfer Protocol).

HTML: gestione di una pagina Web

La pagina HTML può essere composta da una parte di testo, opportunamente disposto, e da immagini di cui si fornisce, all'interno della pagina stessa, l'*URL* (Uniform Resource Locator: l'indirizzo web). In questo modo il browser conosce l'indirizzo cui rivolgersi per richiedere l'immagine da posizionare nella pagina.

La rete Internet è basata sulla *commutazione di pacchetto*: tutto ciò che passa dalla rete è diviso in parti (i pacchetti) di una certa dimensione che, per arrivare a destinazione, possono percorrere strade diverse. Il browser mano a mano che riceve i pacchetti che compongono la pagina da visualizzare, richiede eventuali altre risorse descritte nella pagina e si occupa dell'interpretazione dei tag. La ricezione dei pacchetti in tempi diversi sta alla base di alcuni effetti ben noti a chi naviga in Internet: la pagina, specie se complessa, viene composta non tutta in una unica soluzione, mano a mano che si legge una parte della pagina viene composta quella successiva, le immagini possono arrivare con un certo ritardo e, specie se di grandi dimensioni, possono essere visualizzate anche solo parzialmente.

I tag di HTML sono codificati dal W3C in modo che si possa avere, quanto più possibile una visualizzazione coerente a prescindere dal sistema HW/SW utilizzato e quindi gli sviluppatori di browser sono invitati a rispettare le aspettative grafiche sull'effetto dei tag. Nella realtà la compatibilità dei vari browser è meno generale di quello che ci si aspetta. Spesso gli sviluppatori dei browser, vuoi per fornire maggiori strumenti a chi sviluppa siti web, vuoi per cercare di imporre la propria supremazia nel settore, sono spinti ad arricchire di nuove opzioni i tag o, addirittura, inventarne di nuovi e tutto con conseguenze gravi per chi progetta le pagine web e per chi dette pagine deve visualizzare. Il browser tenta di seguire le direttive espresse dai tag ma se questi non sono riconosciuti vengono, semplicemente, ignorati e quindi il problema della compatibilità non riguarda la possibilità di riscontrare errori, inesistente in HTML, ma la differenza di aspettative fra le direttive dei tag e la reale visualizzazione: in ogni caso la pagina viene visualizzata.

In linea generale il consiglio per chi sviluppa pagine web, per motivi di condivisione e accessibilità più volte evidenziati, è quello di utilizzare i tag definiti e approvati ufficialmente dal W3C.

Del meccanismo che sta alla base della richiesta/ricezione di pagine web è necessario tenere conto nel momento della progettazione delle pagine stesse.

HTML: pagina informativa

L'esempio, proposto come pagina HTML, presenta un brano tratto da un celebre intervento di E. Raymond e, dal punto di vista del layout della pagina, mostra un titolo centrale, una parte di testo formattato utilizzando una tabella, una immagine e la possibilità di visitare altre pagine collegate a questa.

Il sorgente della pagina è riportato di seguito:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset=”UTF-8” />
  <title>pagina dimostrativa</title>
</head>

<body>

  <!-- Questo e' un commento -->

  <div align="center"><h1>La cattedrale e il bazar</h1></div>
  <div align="center">
  <a href="http://www.catb.org/~esr/"><h2>Eric Raymond</h2></a>
  </div>
  <br />

  <!-- Immagine e testo formattati con una tabella -->

  <table>
    <tr>
      <td valign="top">
        <img src="tux.jpg" alt="tux" />
      </td>

      <td>
        ... Linux stravolse gran parte di quel che credevo di sapere. Per
        anni avevo predicato il vangelo Unix degli strumenti agili, dei
        prototipi immediati e della programmazione evolutiva. Ma ero
```

```
        anche convinto che esistesse un punto critico di complessità
        al di sopra del quale si rendesse necessario un approccio
        centralizzato e a priori. Credevo che il software più
        importante (sistemi operativi e strumenti davvero ingombranti
        come <a href="http://www.fsf.org">Emacs</a>) andasse realizzato
        come le cattedrali, attentamente lavorato a mano da singoli geni
        o piccole bande di maghi che lavoravano in splendido isolamento,
        senza che alcuna versione beta vedesse la luce prima del momento
        giusto.
      </td>
    </tr>
  </table>

  <!-- Testo fuori dalla tabella -->

  <div align="left">
  Rimasi non poco sorpreso dallo stile di sviluppo proprio di Linus
  Torvalds: diffondere le release presto e spesso, delegare ad altri
  tutto il possibile, essere aperti fino alla promiscuità. Nessuna
  cattedrale da costruire in silenzio e reverenza. Piuttosto, la
  comunità Linux assomigliava a un grande e confusionario bazar,
  pullulante di progetti e approcci tra loro diversi. Un bazar dal quale
  soltanto una serie di miracoli avrebbe potuto far emergere un sistema
  stabile e coerente. Il fatto che questo stile bazar sembrasse
  funzionare, e anche piuttosto bene, mi colpì come uno shock.
  Mentre imparavo a prenderne le misure, lavoravo sodo non soltanto sui
  singoli progetti, ma anche cercando di comprendere come mai il mondo
  Linux non soltanto non cadesse preda della confusione più totale,
  ma al contrario andasse rafforzandosi sempre più a una
  velocità a malapena immaginabile per quanti costruivano
  cattedrali....
  </div>

</body>
</html>
```

HTML: disposizione degli oggetti nella pagina

Le due dichiarazioni:

```
<!DOCTYPE html>
...
<meta charset="UTF-8" />
```

contengono informazioni sulla corretta interpretazione della pagina. La prima dichiarazione afferma che la pagina è un documento HTML5 e quindi il browser deve prepararsi per l'interpretazione dei tag definita in tale versione. La seconda dichiarazione stabilisce la codifica dei caratteri da usare.

La pagina HTML, racchiusa fra i tag `<html>` e `</html>`, è suddivisa in due parti: l'*intestazione* delimitata da `<head>` e `</head>` e il *corpo* delimitato da `<body>` e `</body>`.

L'*intestazione*, nel caso presentato, contiene una riga delimitata dai tag `<title>` che racchiudono la sequenza di parole che comparirà nella barra del titolo della finestra del browser che visualizza la pagina. Potrebbero essere specificati anche altri *meta tag* utilizzati dai motori di ricerca per indicizzare la pagina. I meta tag, cui si fa riferimento, hanno il formato `<meta name=... content=...>` dove in `name` può essere specificato `keywords`, `description` o `author` ad indicare,

rispettivamente, le parole chiavi sotto le quali indicizzare la pagina, una breve descrizione del contenuto della pagina, il nome dell'autore. La parte `content` serve per specificare, appunto, le parole chiavi, la descrizione, il nome. L'uso nei tag di maiuscole o minuscole o di qualsiasi combinazione fra di esse è ininfluente.

Nel *corpo* della pagina è specificato il testo da visualizzare, il modo come organizzare la pagina e l'indicazione delle altre risorse necessarie. In generale il testo si dispone in modo da occupare la larghezza della finestra del browser. Se la finestra viene ridimensionata, il testo si ridispone all'interno della stessa. Per il resto il modo come il testo si dispone, dipende dai tag presenti. Per esempio la riga:

```
<div align="center"><h1>La cattedrale e il bazar</h1></div>
```

indica che il testo compreso va allineato al centro e visualizzato con un carattere più grande rispetto a quello del resto della pagina. La grandezza dei caratteri è regolata dai tag da `<h1>` (il più grande) ad `<h6>` (il più piccolo). Il carattere di default settato nel browser corrisponde ad `<h3>` che quindi può essere non specificato.

All'interno della pagina sono specificati dei link a risorse esterne:

```
<a href="http://www.catb.org/~esr/"><h2>Eric Raymond</h2></a>
...
<a href="http://www.fsf.org">Emacs</a>
```

se l'utente seleziona con un clic del mouse un'àncora (`Eric Raymond` o `Emacs` rispettivamente), il browser effettua una richiesta all'URL specificato nel tag e la pagina corrispondente è caricata nella finestra.

Subito dopo il titolo del testo da visualizzare è forzata una riga vuota per mezzo del tag `<br />`. Una delle convenzioni che regolano l'uso delle versioni recenti di HTML suggerisce per i tag che non prevedono chiusura (`br` o `img` nell'esempio) l'inserimento del simbolo di chiusura alla fine del tag stesso.

Per formattare il testo in modo che compaia affiancato all'immagine, è stata usata una tabella. La tabella racchiusa fra i tag `<table>` è composta da righe, ognuna racchiusa dai tag `<tr>`, e colonne, ognuna racchiusa dai tag `<td>`.

```
<table>
  <tr>
    <td valign="top">
      <img src="tux.jpg" alt="tux" />
    </td>

    <td>
      ...
    </td>
  </tr>
</table>
```

Nell'esempio è usata una tabella con una sola riga e due colonne. Nella prima cella della tabella si richiede la visualizzazione di una immagine di cui viene fornito l'URL (la specifica `src` del tag `img`) che, in questo caso, si riduce al nome del file che contiene l'immagine stessa ad indicare che, il file, è conservato nello stesso posto della pagina. Il browser per portare a termine tale visualizzazione deve, quindi, a questo punto, richiedere l'invio del file contenete l'immagine specificata.

L'inserimento della specifica `alt` all'interno del tag `img` indica la visualizzazione di un testo alternativo qualora, per un motivo qualsiasi, l'immagine non potesse essere visualizzata. Fra l'altro tale caratteristica è estremamente utile, qualora si scelga un testo alternativo significativo, per permettere una piena comprensione del contenuto della pagina anche da parte, per esempio, di un non vedente.

Se la finestra del browser viene ridimensionata anche la tabella subisce la stessa sorte in modo da contenere il testo previsto compatibilmente, ovviamente, con le dimensioni dell'immagine.

Il successivo testo è inserito, allineato a sinistra, in una nuova division indipendentemente dalla formattazione del sorgente: fra due parole è visualizzato un solo spazio anche se nel sorgente ne sono inseriti più di uno, la suddivisione in righe dipende dai tag usati (per esempio dall'uso del tag `<br />`), la lunghezza della riga dipende dalle dimensioni dell'area di visualizzazione del browser.

La formattazione da HTML ai CSS

I tag trattati rappresentano l'ossatura di HTML ma nella visualizzazione della pagina, anche ai fini di una maggiore comprensione, c'è necessità di avere a disposizione anche altri strumenti di formattazione come, per esempio, la possibilità di utilizzare font e colori diversi per il testo. Effettivamente HTML prevede appositi tag o opzioni da aggiungere. Specialmente le tabelle, utilizzate pesantemente per la costruzione del layout della pagina, prevedono un insieme consistente di opzioni. A titolo di esempio i tag `<table>` o `<td>` prevedono, fra le altre, anche la possibilità di usare l'opzione `width` per mezzo della quale si può specificare la larghezza della tabella/cella che può essere espressa in termini di percentuale rispetto alla finestra del browser o alla tabella (`<table width=70%>`) o direttamente in pixel (`<table width=640>`). Le opzioni che, col passare del tempo sono anche aumentate a dismisura, non sono trattate perché dal momento della progettazione dei CSS tutte le varie opzioni, mano a mano accumulatesi, sono state deprecate nelle versioni recenti di HTML e se ne sconsiglia l'uso a favore dei CSS. Nel 1996 il W3C emana le prime specifiche di CSS (Cascading Style Sheet, in italiano fogli di stile a cascata).

I CSS erano un interessante sistema per separare contenuto da formattazione. La base di questo linguaggio, infatti, consisteva nel fatto che il contenuto sarebbe stato sempre definito dal codice (X)HTML, mentre la formattazione si sarebbe trasferita su un codice ***completamente separato****, il CSS appunto.* (Wikipedia)

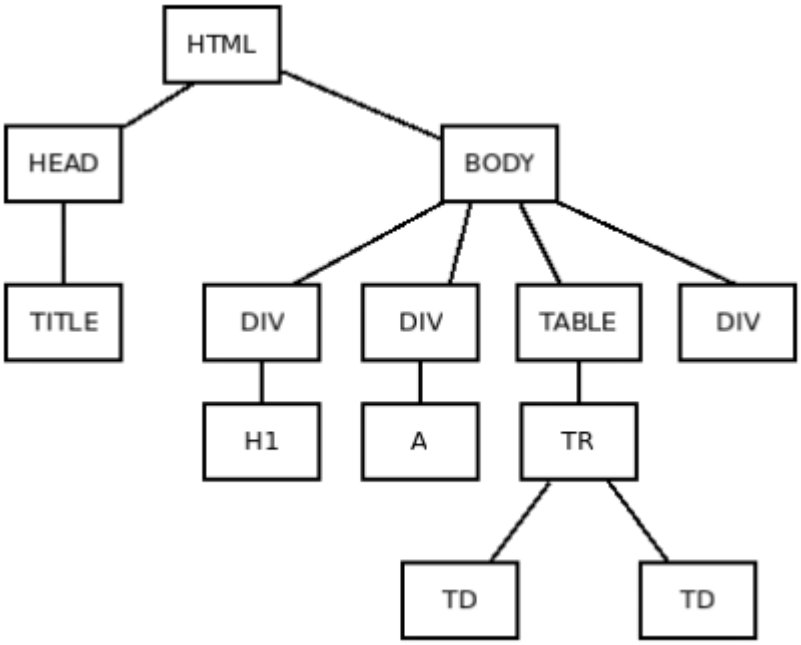
CSS: principi fondamentali

Per comodità si riporta la struttura della pagina HTML esaminata:

```
<html>
<head>
  <title>...</title>
</head>

<body>

  <div align="center"><h1>...</h1></div>
  <div align="center">
  <a href="http://www.catb.org/~esr/">...</a>
  </div>

  <table>
```

```
      <tr>
        <td valign="top">
          ...
        </td>

        <td>
          ...
        </td>
      </tr>
    </table>

    <div align=”left”>
    ...
    </div>

  </body>
  </html>
```

Il consorzio W3C ha fissato nel cosiddetto **Document Object Model** (DOM) lo standard sulla rappresentazione gerarchica di un documento strutturato. Secondo tali indicazioni la pagina HTML si può rappresentare mediante la seguente struttura gerarchica:

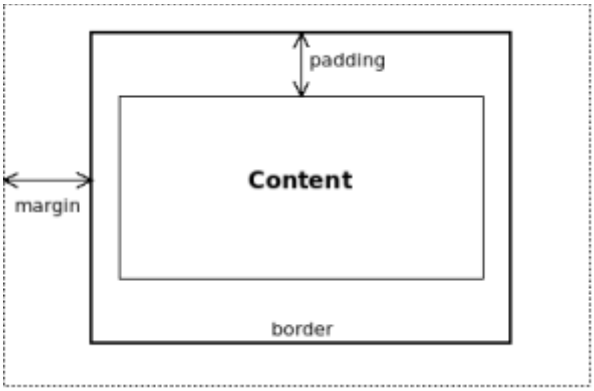


Gli elementi nella pagina hanno una relazione del tipo **genitore-figlio** (**parent-child**). Un elemento è di tipo parent se contiene altri elementi, è di tipo child se è contenuto in un altro elemento. Per esempio TABLE è parent di TR e, a sua volta, child rispetto a BODY. Inoltre BODY è un antenato (**ancestor**) di TD che a sua volta è un discendente (**descendant**) di BODY. L'elemento HTML, che racchiude tutti ma non è racchiuso, si chiama elemento radice (**root**).

Una serie di specifiche attribuite ad un elemento si applicano ai suoi discendenti, a meno di duplicazione diversa della stessa specifica. Per esempio se si specifica per il primo DIV da sinistra il colore rosso per il carattere, tale proprietà varrà per H1 e per altri eventuali elementi gerarchicamente inferiori.

Un concetto fondamentale per comprendere il funzionamento dei CSS è il **box model**. Ogni

elemento di HTML può essere considerato come un riquadro, un box, che contiene del testo. Il box si può posizionare in un punto della pagina e si possono definire le caratteristiche che dovrà avere il testo contenuto.



Nell'immagine sono specificate le caratteristiche del posizionamento del box riferite al box di livello superiore. Per esempio la dimensione totale del box (larghezza = 2*margin + 2*border + 2*padding + content) deve essere non superiore a quella del box contenitore. Il testo vero e proprio è specificato in *Content*, il *Padding* è uno spazio che separa il testo dalla cornice (*Border*), il *Margin* è lo spazio che distanzia il bordo del box dal contenitore di livello superiore.

CSS: tipico layout di sito web

Come esempio di applicazione dei fogli di stile e delle loro interazioni con la pagina HTML, si propone una pagina che adotta un layout abbastanza comune:

La pagina è centrata all'interno della finestra del browser contiene l'intestazione in cui è visualizzato un logo, il corpo centrale suddiviso in tre colonne verticali dedicate all'elenco dei link verso le altre pagine del sito (la colonna di sinistra), al testo della pagina (colonna centrale) e ad un ulteriore elenco di parole chiavi o link a risorse esterne (colonna a destra). In fondo è previsto il piè di pagina dove possono trovare posto indicazioni sull'autore o progettista della pagina ed altro.

Questa descrizione dettagliata è sostanzialmente quello che si vede di conseguenza ad uno sguardo sulla pagina, ma anche quello che si può leggere nel sorgente della pagina HTML:

```
<!DOCTYPE html>
<head>
  <meta charset="UTF-8" />
  <title>Tipico layout sito web</title>

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

<body>

  <!-- contenitore della pagina -->

  <div id="container">
    <div id="header">
      <img src=logo.png />
    </div><!--Header-->

    <!-- contenuto pagina -->

    <div id="content">

      <!-- colonna sinistra -->

      <div class="sidebar primary">
        <ul>
          <li>Hardware</li>
          <li>Software</li>
          <li>Sistemi Operativi</li>
          <li>Periferiche</li>
        </ul>
      </div><!--Sidebar-->

      <!-- colonna centrale -->

      <div id="main">
        <h2>Definizione e storia</h2>
        <p>Un computer (in italiano calcolatore o elaboratore, talvolta
        nello svizzero italiano ordinatore dalla sua denominazione francese
        ordinateur) è una macchina automatizzata in grado di eseguire
        calcoli matematici complessi e, eventualmente, altri tipi di
        elaborazioni di dati. Nato infatti come macchina calcolatrice evoluta,
        a partire dalla seconda metà del XX secolo il computer si evolve in
        macchina in grado di eseguire le più svariate elaborazioni, restituendo
        cioè un certo output a partire da istruzioni impartite in
        input dall'esterno.</p>

        <p>Nel corso della storia, l'implementazione tecnologica di
        questa macchina si è modificata profondamente sia nei meccanismi
        di funzionamento (meccanici, elettromeccanici ed elettronici), che
        nelle modalità di rappresentazione dell'informazione (analogica
        e digitale) che in altre caratteristiche (architettura interna,
        programmabilità, ecc.). Al giorno d'oggi, ci si riferisce comunemente
        al computer come ad un dispositivo elettronico e digitale, programmabile
        a scopo generico, costruito secondo la cosiddetta architettura di
        von Neumann ed il modello teorico-computazionale della cosiddetta
        macchina di Turing. Sebbene i computer programmabili a scopo
```

```
          generico siano oggi i più diffusi esistono in specifici ambiti
          di applicazione modelli di computer dedicati (automazione
          industriale, domotica, ecc.).</p>
        </div><!--Main-->

        <!-- colonna sinistra -->

        <div class="sidebar secondary">
          <ul>
            <li>Linux</li>
            <li>Programmazione</li>
            <li>Wikipedia</li>
          </ul>
        </div><!--Sidebar-->

      </div><!--Content-->

      <!-- fine contenuto pagina -->
      <!-- piè di pagina -->

      <div id="footer">
        testo da Wikipedia
      </div><!--Footer-->

    </div><!--Container-->

  </body>
  </html>
```

Se si legge il codice HTML si può individuare immediatamente la struttura della pagina. I tag `div` con l'identificativo `id` individuano le parti che la compongono: `div` dice che si tratta di un contenitore, `id` con il suo nome autoesplicativo indica il tipo di contenuto di quello specifico contenitore.

La riga:

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

indica che la pagina va collegata (`link`) a un file testo (`type`) che contiene gli stili (`stylesheet`) e di cui si fornisce il riferimento (`style.css`). Il file degli stili contiene le direttive sullo stile da utilizzare per la visualizzazione del testo associato a quel nome (per esempio alla colonna centrale va applicato lo stile `main` la cui definizione si trova in `styles.css`). In questo caso si stanno usando stili *esterni*, specificati cioè in file diverso rispetto a quello che contiene il codice HTML. Gli stili possono essere specificati anche in altri modi (si accennerà in un esempio successivo agli stili inline) ma gli stili esterni sono il sistema più comunemente utilizzato per fare in modo che la parte grafica della pagina (il file CSS) sia distinta dal suo contenuto (il file HTML).

Dal punto di vista HTML gli ulteriori tag di cui fino ad ora non si è trattato si trovano nel frammento:

```
<ul>
  <li>Hardware</li>
  <li>Software</li>
  <li>Sistemi Operativi</li>
  <li>Periferiche</li>
</ul>
...
```

Il tag `ul` (unordered list) indica che si tratta di una lista i cui elementi sono specificati dai tag `li`. HTML può prevedere, per ogni elemento della lista, la presenza di un punto elenco (che può essere anche un quadratino o una immagine) che precede la voce. Possono essere utilizzate anche liste ordinate (tag `ol`) i cui elementi (sempre `li`) sono preceduti da un numero progressivo arabo o latino o una lettera.

CSS: definizione degli stili

Il file `styles.css` contiene gli stili da applicare alla pagina HTML. Un foglio di stile è composto da una serie di regole. Ogni regola è costituita da un blocco:

Selettore { Dichiarazione ; Dichiarazione ; ...}

il selettore fa riferimento all'elemento HTML cui si vuole applicare lo stile. Il blocco delimitato dalle parentesi graffe racchiude le dichiarazioni che possono essere in qualsiasi numero. Generalmente si preferisce, per motivi di chiarezza, scrivere una dichiarazione per riga. Ogni dichiarazione è composta dalla coppia:

Proprietà : valore

è possibile tuttavia utilizzare una forma semplificata, di cui si esamineranno diversi esempi, che mette assieme più valori riferitesi a diversi aspetti della stessa proprietà. Per esempio il `padding` può essere diviso in più parti (quattro): `padding-top`, `right`, `bottom` o `left` e assegnare valori per ciascuna proprietà, oppure si possono assegnare i valori, separati da spazio, nell'unica proprietà `padding`.

I selettori possono fare riferimento direttamente a tag HTML come:

```
body {
  ...
}
```

oppure nomi, preceduti da `#`, che vanno associati al tag `div` mediante la specifica `id` come:

```
#content {
  ...
}
...
<div id="content">
```

oppure ancora nomi preceduti dal punto e, in questo caso, vanno applicati per mezzo della specifica `class`:

```
#content .sidebar.primary {...}
...
<div class="sidebar primary">
```

nel caso esposto lo stile si applica alla classe `sidebar primary` del box `content`. Così come:

```
#content #main h2 {...}
```

si applica al tag `h2` del box `main`, a sua volta contenuto nel box `content`.

Per esporre le regole di composizione si esamineranno ora gli stili definiti (il testo racchiuso fra `/*` e `*/` è un commento), nel file `styles.css`, per la pagina HTML. Quando è utile aggiungere

commenti più estesi, che possono essere più chiarificatori, il listing del file viene interrotto:

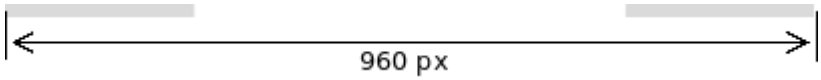
```
body {
  background: #7c7c7c url(body-bg.png);
  font-family: Helvetica, Arial, Sans-Serif;  /* font da utilizzare */
}
```

il primo stile riguarda il corpo della pagina (il tag `body`). La prima dichiarazione setta due proprietà dello sfondo: il colore e una immagine, di cui si fornisce l'url, che ricoprirà l'area. In questo caso alla proprietà (`background`) sono associati due valori separati da uno spazio: si tratta di una forma ridotta permessa per raggruppare più valori che altrimenti dovrebbero essere divisi uno in ogni dichiarazione diversa. L'immagine qui, essendo lo sfondo ricoperto da un colore uniforme, ha dimensioni minime (1,4K) e verrà ripetuta in orizzontale e verticale in modo da coprire tutta l'area.

Nella seconda riga si specificano le famiglie di font da utilizzare. Se il browser non è in grado di utilizzare il primo, si passa al secondo e così via. Altre famiglie utilizzabili potrebbero essere `Serif` (fonte con grazie), `Monospace`.

```
/* contenitore della pagine */

#container {
  width: 960px;     /* larghezza in pixel del box */
  margin: 0 auto;   /* il browser sistema il box rispetto alla finestra */
  background: #fff; /* codice esadecimale del colore */
                    /* si può ricavare anche usando Gimp */
}

/* intestazione */

#header {
  height: 70px;                 /* altezza in pixel dell'area */
  padding: 30px 20px 20px 30px; /* forma semplificata per specificare */
                                /* distanza nell'ordine top right */
                                /* bottom left */
  background: url(header.png);  /* stesse considerazioni del precedente */
                                /* body-bg.png */
}

/* contenitore della pagina */

#content {
  overflow: auto;
  background: url(faux-columns.png) repeat-y;
}
```

la prima proprietà di `content` serve ad avvisare il browser che deve occuparsi di allungare l'area qualora il testo non potesse essere contenuto in essa (gestire l'`overflow`). L'immagine di background (`faux-columns.png`), a differenza di quella usata come sfondo della pagina o dell'intestazione (`body-bg.png` o `header.png`), è una sottile striscia della lunghezza del box `container` (che contiene il box `content`) con i colori delle tre colonne:



la specifica `repeat-y` serve per replicare l'immagine per tutta l'altezza del box.

```
#content .sidebar {          /* caratteristiche generali colonne laterali */
```

```
  width: 180px;              /* larghezza in pixel */
  float: left;               /* posizionamento a sinistra */
                             /* altri box posizionati dopo */
}
#content .sidebar.primary { padding: 0 20px 20px 30px; }
#content .sidebar.secondary { padding: 0 30px 20px 20px; }

#content .sidebar li {
  list-style: none;     /* nessun punto elenco. Poteva essere uno fra: */
                        /* circle square upper-roman lower-alpha */
                        /* oppure list-style-image: url(...); */
  font-size: 16px;      /* dimensione font da utilizzare nei punti elenco */
  color: #666;
  line-height: 24px;    /* spaziatura fra le righe */
}

/* colonna centrale */

#content #main {
  width: 460px;         /* larghezza contenuto */
  float: left;
  padding: 0 20px 20px 20px;
}
#content #main h2 {
  font: italic 30px Georgia, Serif;  /* font e dimensione */
  color: #aaa;
  margin: 0 0 10px 0;
}
#content #main p {
  font-size: 14px;
  color: #666;
  line-height: 24px;
  margin: 0 0 15px 0;
}

/* piè di pagina */

#footer {
  height: 30px;
  background: url(header.png);
  font-size: 12px;
  color: #666;
  padding: 5px 0 0 15px;
}
```

Siti statici e siti dinamici

Le pagina HTML consentono una interazione, con l'utente che la visualizza, molto limitata: l'unica cosa che si può fare è scegliere di seguire i link previsti all'atto della progettazione delle pagine. Si parla in questo caso di **siti statici**: il contenuto delle pagine è stabilito una volte per tutte nel momento della progettazione. Per la prima volta nel Marzo 1995 il numero di siti .com (siti commerciali) supera quello dei siti .edu (siti universitari e accademici in generale). Si rende necessario far diventare il sito web qualcosa in più di una *vetrina espositiva* di prodotti per fargli assumere il compito di interfaccia utente/azienda. HTML viene espanso aggiungendo i `form` che consentono di inserire nella pagina web i controlli tipici presenti nelle interfacce grafiche (pulsanti,

menù di scelta, campi di inserimento ecc...) in modo da permettere l'inserimento di dati. Ci dovrà essere poi qualcosa che avrà il compito di gestire questi dati.

Lo sviluppo dei linguaggi di scripting registra una notevole accelerazione. Per le applicazioni web si distinguono linguaggi di scripting client-side (le istruzioni che compongono il programma sono eseguite nel client) e linguaggi di scripting server-side (istruzioni del programma eseguite nel server) che interagiscono con HTML accedendo ai dati inserite per mezzo dei form, elaborandoli e costruendo, in dipendenza delle elaborazioni, una pagina HTML dipendente dai risultati ottenuti (**siti dinamici**). Una applicazione web, per la costruzione di un sito web dinamico, prevede l'utilizzo di linguaggi dei due tipi.

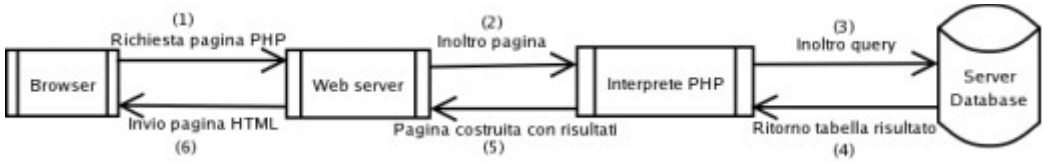
Perché linguaggi di scripting? Si tratta di linguaggi interpretati e questo è un fattore importante in una applicazione di rete per due ordini di motivi: la trasportabilità (le piattaforme presenti in una rete sono molto diverse fra di loro e un programma scritto in un linguaggio interpretato non fa riferimento alla piattaforma ma richiede solo la presenza di un interprete), la sicurezza (un programma interpretato gira *dentro* l'ambiente dell'interprete e non può accedere a zone che possono creare malfunzionamenti).

Perché client-side e server-side? In una rete un server deve offrire i propri servizi a quanti più client possibili e a quanta più velocità possibile. La velocità dipende anche dal traffico presente sulla rete ed è importante occuparla il minor tempo possibile come è importante tenere impegnato il server per il minor tempo possibile. Tutto ciò porta ad una conclusione:

> Tutte le elaborazioni possibili devono essere svolte nel client. Si chiede l'intervento del server quando indispensabile.

Lo sviluppo di una applicazione web richiede quindi l'utilizzo di linguaggi di entrambi i tipi: il compito principale è stabilire *chi deve fare cosa*. Per quali elaborazioni usare un linguaggio client-side e per quali uno server-side.

Dal punto di vista del client il browser diventa, anche, un interprete che è in grado di elaborare le istruzioni di uno script in linguaggio JavaScript. Lo script può essere *embedded* (inglobato nella pagina HTML) o in un file a parte, richiamato da un apposito tag, da HTML.

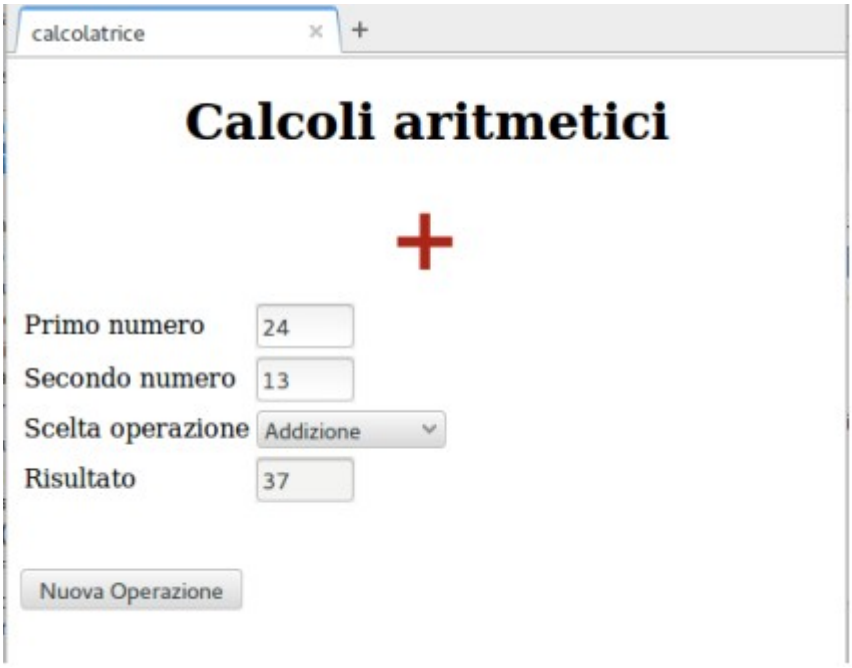


Dal punto di vista del server, il web server interagisce con un programma, per esempio in PHP, che effettua l'interrogazione del database dell'azienda di seguito alle richieste inoltrate dall'utente attraverso un form, e costruisce la pagina HTML contenete i risultati della query. La pagina verrà successivamente inviata al richiedente a cura sempre del web server.

I linguaggi di scripting, a differenza di HTML che è un linguaggio di definizione della pagina, sono linguaggi di programmazione e le pagine HTML diventano lo strumento di interfaccia con l'utente che permette di acquisirne le richieste laddove, invece, l'elaborazione è effettuata in accordo alle istruzioni dello script.

JavaScript: pagina web con semplice calcolatrice

L'esempio proposto di seguito permette un primo livello di interazione con l'utente. C'è la possibilità di immettere dei valori e scegliere una delle operazioni aritmetiche previste.



La casella di scelta permette di selezionare l'operazione da effettuare sui valori inseriti. Se si sceglie una nuova operazione, la pagina *reagisce* modificando la grafica sotto il titolo e visualizzando nel campo `Risultato` il valore calcolato. Il pulsante `Nuova Operazione` permette l'azzeramento dei campi, riportando il modulo alla situazione di partenza con tutti i campi azzerati, e la *scomparsa* del simbolo grafico dell'operazione. Per motivi di chiarezza e per mettere in evidenza le istruzioni JavaScript, il codice HTML è ridotto al minimo e non sono usati CSS.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8" />
  <title>calcolatrice</title>

  <script>

    // precaricamento immagini

    var operaz    = new Array();
    operaz[0]     = new Image();
    operaz[0].src = "blank.png";
    operaz[1]     = new Image();
    operaz[1].src = "piu.png";
    operaz[2]     = new Image();
    operaz[2].src = "meno.png";
    operaz[3]     = new Image();
    operaz[3].src = "per.png";
    operaz[4]     = new Image();
    operaz[4].src = "diviso.png";

    // Calcolo risultato operazione aritmetica
```

```
    function calcola(){

      var indice = document.f1.operazione.selectedIndex;
      var n1     = eval(document.f1.primo.value);
      var n2     = eval(document.f1.secondo.value);

      switch (indice){
        case 1:
          document.f1.risultato.value = n1 + n2;
          break;
        case 2:
          document.f1.risultato.value = n1 - n2;
          break;
        case 3:
          document.f1.risultato.value = n1 * n2;
          break;
        case 4:
          document.f1.risultato.value = n1 / n2;
          break;
      }

      document.simbolo.src=operaz[indice].src;
    }

    // Reset modulo

    function azzera(){
      document.f1.reset();
      document.simbolo.src=operaz[0].src;
    }
  </script>

</head>
<body>
  <div align="center"><h1>Esempio calcoli aritmetici</h1></div>
  <br>
  <div align="center"><IMG src="blank.png" name="simbolo" alt="simbolo
  operazione"></div>
  <br>

  <!-- Acquisizione degli input dell'utente -->

  <form name="f1">
  <table>
    <tr>
      <td>Primo numero</td>
      <td><input type="text" name="primo" size="5"></td>
    </tr>

    <tr>
      <td>Secondo numero </td>
      <td><input type="text" name="secondo" size="5"></td>
    </tr>

    <tr>
      <td>Scelta operazione</td>
      <td>
        <select name="operazione" onChange="calcola()">
```

```
            <option>
            <option>Addizione
            <option>Sottrazione
            <option>Moltiplicazione
            <option>Divisione
          </select>
        </td>
      </tr>

      <tr>
        <td>Risultato</td>
        <td><input type="text" name="risultato" size="5" readonly=”readonly”></td>
      </tr>
    </table>
    <br><br>

    <input type="button" value="Nuova Operazione" onClick="azzera()">

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

JavaScript: intercettare gli eventi

HTML permette l'introduzione di dati da input per mezzo dell'utilizzo dei moduli (`form`):

```
<form name="f1">
  ...
  Primo numero
  <input type="text" name="primo" size="5">
  ...
  Secondo numero
  <input type="text" name="secondo" size="5">
  ...
  Scelta operazione
  <select name="operazione" onChange="calcola()">
    <option>
    <option>Addizione
    <option>Sottrazione
    <option>Moltiplicazione
    <option>Divisione
   </select>
   ...
   Risultato
   <input type="text" name="risultato" size="5" readonly=”readonly”>
   ...
   <input type="button" value="Nuova Operazione" onClick="azzera()">
 </form>
```

In questo caso il modulo, cui è stato dato il nome `f1`, comprende:

- tre input di tipo testo: due per l'input dei numeri su cui effettuare l'operazione (con nome `primo` e `secondo`) sui quali l'utente può inserire valori, uno (con nome `risultato`) per la visualizzazione dei risultati che prevede l'opzione `readonly=”readonly”` per non permettere l'inserimento di dati da parte dell'utente: è utilizzato come campo di output. La specifica `size` indica la lunghezza visiva della casella di inserimento.

- una casella di selezione, chiamata `operazione`, per mezzo della quale si può scegliere il tipo di operazione da effettuare. La prima opzione, in questo caso vuota, è quella selezionata per default: manca infatti, in una qualsiasi delle `option`, la clausola `selected` che la farebbe scegliere come opzione selezionata di default
- un pulsante (input di tipo `button`) per l'azzeramento del modulo e il ripristino delle condizioni di avvio della pagina.

Sia sulla casella di selezione che sul pulsante si vogliono intercettare due eventi: `onChange`, per la casella di selezione, che intercetta l'azione dell'utente di scegliere una opzione diversa rispetto a quella visualizzata in questo momento e `onClick`, per il pulsante, che intercetta il click del mouse sul pulsante stesso. Alla pagina vengono quindi aggiunte capacità di rispondere a determinati eventi (*event driven*).

In ambedue i casi l'evento avvia un opportuno gestore (*event handler*): la funzione `calcola()` nel caso della modifica dell'opzione selezionata, la funzione `azzera()` nel caso della pressione del pulsante.

JavaScript: accesso agli oggetti della pagina

JavaScript permette di vedere la pagina, individuata dal nome `document`, come un *contenitore* al cui interno si trovano gli oggetti della pagina (form, pulsanti, caselle di testo, ma anche immagini, links) cui è possibile accedere, per modificarne le proprietà, per mezzo del nome. Per esempio `document.f1.primo.value` o `document.f1.secondo.value` permettono di conoscere quello che ha digitato l'utente della pagina nelle rispettive caselle di testo. Letteralmente è come se si dicesse, per esempio, il valore (`value`) di `primo` che fa parte di `f1` che, a sua volta, è contenuto nella pagina corrente (`document`). Il carattere `.` distingue i vari livelli gerarchici.

`document.f1.operazione.selectedIndex` permette, invece, di conoscere l'opzione selezionata. Il valore sarà 0 per la prima opzione, 1 per la seconda ecc...

Si possono anche *inviare dei messaggi ad un oggetto* se questo è abilitato a ricevere quel tipo di messaggio. Per esempio `document.f1.reset()` invoca il metodo `reset()` di `f1`: invia al form il messaggio e questi risponde reinizializzando i campi del modulo.

Sono le *proprietà della programmazione ad oggetti*: è definita, nella pagina, una gerarchia di oggetti a partire dall'oggetto `document` che, negli esempi, rappresenta il livello più alto. Sotto `document` ci sono i `form` che, a loro volta, hanno sotto i singoli elementi che li compongono, i links, le immagini e così via.

`Array` o `Image` sono dei tipi (*classi*) che mettono a disposizione, qualora si definisca un oggetto di quel tipo, la modifica di determinate proprietà:

```
var operaz    = new Array();
operaz[0] = ...
operaz[1] = ...
...
```

In questo caso `operaz` viene definito come oggetto del tipo `Array`. Poiché, poi, un `Array` è una variabile con indice, si può assegnare un valore ad ogni singolo elemento identificato dall'indice.

Anche con `Image` si possono generare oggetti cui possono essere modificate certe proprietà:

```
operaz[0]     = new Image();
```

```
operaz[0].src = "blank.png";
```

In questo caso, per prima cosa, nella posizione 0 dell'array `operaz` viene inserito un oggetto di tipo immagine, poi, si modifica l'attributo `src` (*source*) dell'immagine mettendo il nome del file che contiene l'immagine stessa. Da questo momento in poi `operaz[0]` è una variabile di memoria con l'immagine che è conservata nel file `blank.png`.

JavaScript: gestione degli eventi

Tutto il codice JavaScript contenuto nella pagina è posto, in questo caso, in un unico blocco delimitato dai tag `<script>` e `</script>` posto nell'intestazione della pagina.

La parte di codice comprendente la generazione di `operaz[]` viene eseguita non appena ricevuta, da parte del browser, la parte di pagina che la contiene. Le funzioni `calcola()` e `azzera()` sono gestori di eventi e verranno, invece, eseguite in risposta agli eventi prima specificati.

```
// Calcolo risultato operazione aritmetica

function calcola(){

  var indice = document.f1.operazione.selectedIndex;
  var n1     = eval(document.f1.primo.value);
  var n2     = eval(document.f1.secondo.value);

  switch (indice){
    case 1:
      document.f1.risultato.value = n1 + n2;
      break;
    case 2:
      document.f1.risultato.value = n1 - n2;
      break;
    case 3:
      document.f1.risultato.value = n1 * n2;
      break;
    case 4:
      document.f1.risultato.value = n1 / n2;
      break;
  }

  document.simbolo.src=operaz[indice].src;
}
```

Ogni modifica dell'opzione selezionata (evento `onChange`) lancia l'esecuzione del gestore.

Innanzitutto sono dichiarate tre variabili contenenti il numero d'ordine dell'opzione selezionata (la variabile `indice`) e i valori numerici degli input dell'utente (la funzione `eval` fa diventare numeri quelli che vengono acquisiti come testo). A questo punto la maggior parte del lavoro è fatta: basta individuare quale è l'operazione scelta (la struttura `switch/case` successiva) e assegnare il valore a `risultato`. Questo garantirà la visualizzazione del numero opportuno nella casella di testo utilizzata come output nel modulo.

L'ultima operazione `document.simbolo.src=operaz[indice].src` consente di modificare il sorgente dell'immagine `simbolo` contenuta nella pagina, che inizialmente è `blank.png`, cioè un quadratino vuoto, con l'immagine rappresentante l'operazione effettuata. Per esempio `operaz[1].src` è `piu.png` ovvero il simbolo di somma aritmetica.

```
// Reset modulo

function azzera(){
  document.f1.reset();
  document.simbolo.src=operaz[0].src;
}
```

L'evento `onClick` sul pulsante `Nuova Operazione` lancia il gestore `azzera()`. Il gestore azzera il modulo (`document.f1.reset()`) e cambia il sorgente di `simbolo` riportandolo a `blank.png` cioè al quadratino vuoto e, quindi, visivamente, si nota la *scomparsa* del simbolo dell'operazione.

In verità se non fosse stato per quest'ultima esigenza, il *reset* del modulo poteva essere fatto in maniera più semplice definendo il pulsante non genericamente come tipo `button`, ma come tipo `reset`. In quest'ultimo caso sarebbe bastato solo il click del mouse sul pulsante per azzerare il modulo.

Problematiche di una elaborazione client-side

JavaScript è un linguaggio, principalmente, *client-side*: le elaborazioni avvengono direttamente nel client, il server si limita a spedire la pagina.

Si è già precedentemente accennato al fatto che, in linea generale, una pagina web arriva al client suddivisa in pacchetti che possono giungere anche in momenti diversi. Tutto ciò può comportare un funzionamento anomalo della pagina qualora, per esempio, l'utente della pagina di esempio cambi il tipo di operazione da effettuare, senza che però la parte di pagina contenete il gestore dell'evento sia ancora arrivata al client e, quindi, senza che l'operazione scelta possa essere effettuata. Questo problema può essere risolto facilmente inserendo il codice del gestore dell'evento in una parte della pagina che *sicuramente* arriverà prima della parte contenente il form. L'intestazione è il posto più opportuno per ospitare i gestori di eventi essendo la prima parte della pagina stessa: nessun elemento della pagina viene visualizzato (si tenga presente che la parte visualizzata è contenuta nel corpo della pagina) se prima non è arrivata l'intestazione.

Un altro problema di cui tenere conto è legato alla manipolazione delle immagini contenute nella pagina.

Le immagini sono inviate, dal server, in seguito a richiesta proveniente dal browser, nel momento in cui ci si accorge che sono necessarie: questo è il motivo per cui, a volte, le immagini possono venire visualizzate con un certo ritardo rispetto al testo che compone la pagina. Tutte le componenti della pagina, quando ricevute dal server, vengono conservate nella *cache* (una parte di memoria di massa allo scopo riservata) da cui vengono prelevate per essere visualizzate. Quando si richiede una nuova pagina, questa viene visualizzata con un certo ritardo dipendente dalla velocità della connessione e dalla *pesantezza* della pagina (la dimensione complessiva di tutte le componenti della pagina), ritardo che non si percepisce quando invece si carica una pagina precedentemente visualizzata. In questo caso, infatti, la pagina viene recuperata dalla cache.

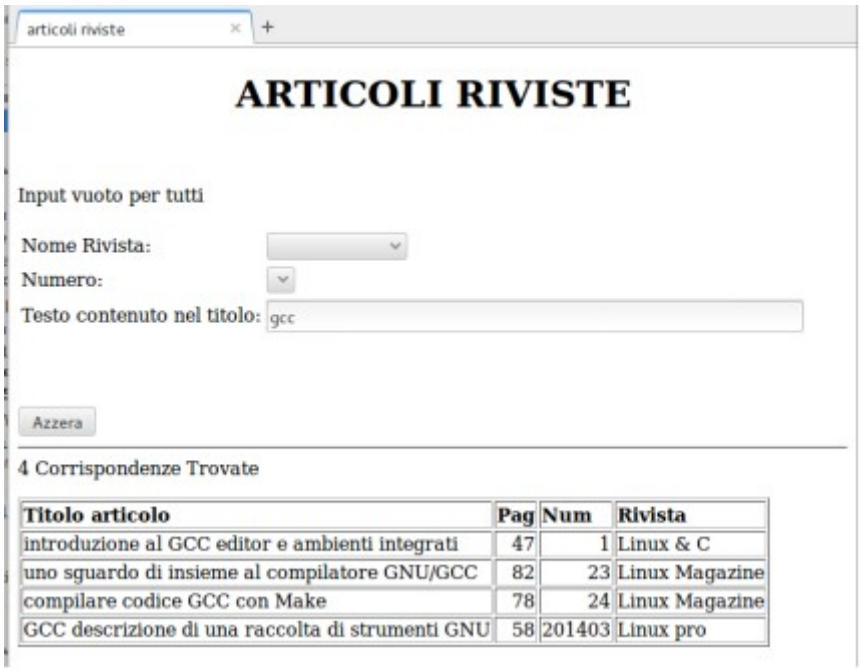
Nell'esempio proposto la scelta dell'operazione, oltre che un nuovo calcolo, comporta una modifica del simbolo grafico. Tale modifica, affinché produca l'effetto voluto, è necessario sia immediata: non si può aspettare che arrivi dal server l'immagine dell'operazione richiesta con tutto quello che ciò significa in termini di ritardo. Si è fatto notare, in precedenza, che l'assegnazione del nome dei file che contengono le immagini, quindi il caricamento delle immagini nella cache, veniva eseguita immediatamente, la modifica dell'immagine visualizzata, invece, veniva eseguita, come giusto, in

corrispondenza dell'evento. La strutturazione del codice nella pagina di esempio, garantisce l'immediatezza della modifica perché l'immagine viene recuperata dalla cache e non viene richiesta quando necessaria per la visualizzazione.

Ovviamente se si prova la pagina in locale tutte le osservazioni fatte in precedenza non hanno alcun valore, se però si opera in un ambiente di rete, a maggior ragione se la pagina risiede in un computer remoto, la posizione del codice JavaScript all'interno della pagina e il precaricamento delle immagini possono fare la differenza tra una pagina funzionante come ci si attende ed una non perfettamente rispondente alle aspettative.

PHP: costruzione di pagine dinamiche

Si presenta ora come esempio di costruzione di pagine da parte di uno script PHP un'applicazione che permette la ricerca di articoli pubblicati in riviste.



l'utente può scegliere il nome della rivista di cui visualizzare gli articoli pubblicati, il numero della rivista (che sarà disponibile in seguito alla scelta del nome) o una parola che dovrà essere contenuta nel titolo degli articoli che interessano. L'interazione con l'utente è gestita in modo che la tabella dei risultati è visualizzata o aggiornata tutte le volte che si sceglie una rivista diversa, un numero diverso o si inserisce un testo nella casella del titolo. In questo ultimo caso è necessario portare il cursore di inserimento fuori dal controllo (clic sulla pagina per esempio) in modo che il codice *si accorga* che c'è stato un cambiamento. Se non ci sono scelte (la prima volta che si esegue la pagina o in seguito alla pressione del tasto `Azzera` quando tutti i campi di inserimento sono vuoti) la parte inferiore della pagina con la tabella dei risultati, non viene visualizzata.

Nel database `gestione` cui si fa riferimento nella pagina sono definite due tabelle: `riviste(id, nome)` e `articoli(id, titolo, pag, num, id_rivista)`. Gli articoli sono collegati alle riviste che li pubblicano per mezzo della chiave esterna `id_rivista` che fa riferimento alla chiave primaria `id` della tabella `riviste`.

Il codice della pagina (il nome della pagina è `pagina.php`) è un po' complesso ed è scritto tenendo

come punto di riferimento l'interazione con l'utente più che l'aspetto estetico. Sono utilizzati i tag HTML indispensabili ma sono presenti righe di codice JavaScript, un accenno ai CSS e, naturalmente e principalmente, PHP. Il codice JavaScript e PHP coesistono all'interno della stessa pagina HTML ma, naturalmente, è diverso il *quando* viene eseguito e il *chi* lo esegue. **All togheter now!**

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8" />
  <title>articoli riviste</title>

  <script>

    function richiama(){
      document.f1.submit.click();
    };

    function azzera(){
      document.f1.nome.value='';
      document.f1.numero.value='';
      document.f1.ttit.value='';
      richiama();
    };

  </script>

</head>
<body>
  <div align=center><h1>ARTICOLI RIVISTE</h1></div>

  <?php
    $connessione = mysql_connect("localhost","root","root");
    mysql_select_db("gestione");
    $q = "SELECT * FROM riviste ORDER BY nome";
    $recset = mysql_query($q);
  ?>

  <br /><br />Input vuoto per tutti<br /><br />

  <!-- interazione con l'utente -->

  <form method="post" name="f1" action="pagina.php">
    <table>
      <tr>
        <td>Nome Rivista: </td>
        <td>

          <!--  Selezione nome della rivista -->

          <select name="nome" onChange=richiama()>
            <option>

            <?php
              while($tr = mysql_fetch_array($recset)){
                if($_POST[nome]==$tr[nome])
                  echo "<option selected> $tr[nome]";
                else
```

```
                    echo "<option> $tr[nome]";
                };
              ?>
            </select>

          </td>
        </tr>
        <tr>
          <td>Numero: </td>
          <td>

            <!-- Selezione numero della rivista scelta -->

            <select name="numero" onChange=richiama()>
              <option>

              <?php
                if(!empty($_POST[nome])){
                  $q2  = "SELECT DISTINCT num FROM riviste,articoli ";
                  $q2 .= "WHERE riviste.id=articoli.id_rivista AND ";
                  $q2 .= "riviste.nome='$_POST[nome]' ";
                  $q2 .= "ORDER BY num DESC";

                  $recset2 = mysql_query($q2);
                  while($tr2 = mysql_fetch_array($recset2)){
                    if($_POST[numero]==$tr2[num])
                      echo "<option selected> $tr2[num]";
                    else
                      echo "<option> $tr2[num]";
                  };
                };
              ?>
            </select>

          </td>
        </tr>
        <tr>
          <td>Testo contenuto nel titolo: </td>
          <td>
            <?php
              echo "<input type=text name=ttit size=50 value='$_POST[ttit]' “;
              echo “onChange=richiama()>";
            ?>
          </td>
        </tr>
      </table>

      <p style="visibility : hidden;">
        <input type="submit" name="submit">
      </p>
      <input type="button" value="Azzera" onClick=azzera()>
    </form>

    <?php

    // se ricerca avviata

    if($_POST[nome] || $_POST[numero] || $_POST[ttit]){
      echo "<hr />";
```

```
    $q  = "SELECT titolo,pag,num,riviste.nome ";
    $q .= "FROM articoli,riviste ";
    $q .= "WHERE articoli.id_rivista=riviste.id ";

    // se c'e' il nome della rivista

    if($_POST[nome]){
      $q .= "AND riviste.nome='$_POST[nome]' ";
    };

    // se c'e' il numero della rivista

    if($_POST[numero]){
      $q .= "AND articoli.num ='$_POST[numero]' ";
    };

    // se c'e' il testo contenuto nel titolo dell'articolo

    if($_POST[ttit]){
      $q .= "AND articoli.titolo LIKE '%$_POST[ttit]%'";
    };

    // risultati ordinati

    $q .= "ORDER BY riviste.nome, articoli.num, articoli.pag";

    // query e risultati

    $recset = mysql_query($q);
    $numrec = mysql_num_rows($recset);

    echo “$numrec Corrispondenze Trovate<br /><br />";

    if($numrec){
      echo "<table border='1'><tr><td><b>Titolo articolo</b></td>";
      echo "<td><b>Pag</b></td>";
      echo "<td><b>Num</b></td>";
      echo "<td><b>Rivista</b></td></tr>";

      while($tr = mysql_fetch_array($recset)){
        echo "<tr>";
        echo "<td> $tr[titolo] </td>";
        echo "<td align='right'> $tr[pag] </td>";
        echo "<td align='right'> $tr[num] </td>";
        echo "<td> $tr[nome] </td>";
        echo "</tr>";
      };

      echo "</table>";
    }; // fine tabella risultati

  }; // fine verifica ricerca avviata

  mysql_close($connessione);
?>

</body>
</html>
```

PHP: interazione con la pagina

Affinché un form HTML sia in condizioni di poter inviare dati ad un server remoto, sono necessarie delle specifiche suppletive:

```
<form method="post" name="f1" action="pagina.php">
  ...
  <select name="nome" onChange=richiama()>
    <option>
    ...
  </select>
  ...
  <p style="visibility : hidden;">
    <input type="submit" name="submit">
  </p>
  <input type="button" value="Azzera" onClick=azzera()>
</form>
```

- l'attributo `method` indica il modo con cui i dati saranno trasmessi al server. I valori possibili sono due: `GET` per accodare i dati alla richiesta della pagina, `POST` per *impacchettarli* assieme alla richiesta. In generale si preferisce il secondo metodo perché permette di mandare una quantità maggiore di dati e, inoltre, i dati stessi non sono visibili nella riga di richiesta pagina
- l'attributo `action` indica l'URL della risorsa che deve processare i dati. In questo caso si tratta dello stesso file PHP
- il pulsante di tipo `submit` consente, se premuto, di inviare i dati ed avviare il processo per il loro trattamento

I dati inviati per mezzo di un form, sono accessibili in PHP per mezzo dell'utilizzo di un *array associativo*. Si tratta di un array particolare i cui elementi, invece che con un indice numerico, sono accessibili per mezzo di un nome. Nel caso del frammento riportato immediatamente sopra, il dato del nome è accessibile utilizzando `$_POST[nome]` (le variabili in PHP sono precedute dal simbolo `$`). Il nome `$_POST[]` è legato al metodo utilizzato per la trasmissione dei dati: se fosse stato `GET`, l'array a cui fare riferimento sarebbe stato `$_GET[]`.

L'automatismo della pagina consente di inviare i dati per l'elaborazione in maniera trasparente per l'utente. Il pulsante di tipo `submit` esiste per permettere l'invio dei dati, ma non è visibile e viene attivato da apposito codice.

```
<p style="visibility : hidden;">
  <input type="submit" name="submit">
</p>
```

nel frammento di codice viene utilizzato **CSS inline**: invece di fare riferimento ad un file esterno con la definizione degli stili come in precedenza, qui le proprietà sono specificate direttamente nel tag interessato: il tag `p` di paragrafo. Questa tecnica può essere utilizzata quando si tratta di definire uno stile che vale solo in un caso, come nell'esempio, e se fosse usata in maniera estesa non ci sarebbero i vantaggi di avere distinti il contenuto della pagina dal layout.

Tutti i controlli prevedono l'intercettazione dell'evento legato al cambiamento del valore nel controllo:

```
function richiama(){
```

```
  document.f1.submit.click();
};
...
<select name="nome" onChange=richiama()>
...
<select name="numero" onChange=richiama()>
...
<input type=text name=ttit ... onChange=richiama()>
```

In ogni controllo viene intercettato il cambiamento del valore contenuto e viene avviato il gestore `richiama()`. Per l'input di tipo `text` affinché si possa generare l'evento è necessario portare il cursore fuori (fin quando resta dentro il controllo l'inserimento non è concluso).

La funzione `richiama()` non fa altro che effettuare il click sul pulsante `submit`. La pagina viene richiamata (l'`action` del `form`) e i dati contenuti nei controlli sono passati per mezzo dell'array `$_POST[]`.

```
function azzera(){
  document.f1.nome.value='';
  document.f1.numero.value='';
  document.f1.ttit.value='';
  richiama();
};
...
<input type="button" value="Azzera" onClick=azzera()>
```

La pressione sul pulsante `Azzera` genera un evento gestito da `azzera()` che inizializza a valore vuoto i controlli e richiama nuovamente la pagina.

PHP: query e costruzione nuova pagina

Le parti di codice PHP che gestiscono l'elaborazione sui dati immessi, possono comparire in qualunque parte della pagina, purché racchiusi dentro i delimitatori `<?php` e `?>`. Si noti che non si tratta, in questo caso di un tag di apertura e di un tag di chiusura, ma, si potrebbe dire, di un unico tag che racchiude il frammento di codice. Il codice PHP può essere miscelato con HTML in qualsiasi modo si ritenga utile e comodo. PHP è *case-sensitive* cioè distingue fra caratteri maiuscoli e minuscoli e quindi, per esempio, `$_POST` va scritto utilizzando lettere maiuscole altrimenti non viene riconosciuto.

PHP mette a disposizione funzioni per interfacciarsi a parecchi DB server. Quelle, per esempio, che consentono di interagire con un server mySQL cominciano tutte per `mysql_`.

```
<?php
  $connessione = mysql_connect("localhost","root","root");
  mysql_select_db("gestione");
  $q = "SELECT * FROM riviste ORDER BY nome";
  $recset = mysql_query($q);
?>
```

In questo frammento di codice viene effettuata la connessione (`mysql_connect`), selezionato il database su cui si vuole accedere (`mysql_select_db`), costruita una query e depositata in una variabile (`$q`) dopo di che viene effettuata la query (`mysql_query`). Il risultato della query viene depositato in `$recset`.

La connessione viene chiusa nella riga di codice:

```
mysql_close($connessione);
```

La query può anche essere più complessa e può essere utile, per ragioni di leggibilità, costruirla pezzo per pezzo. In questi casi può essere di aiuto l'operatore `.=` che accoda la stringa specificata a destra alla variabile di sinistra, così come, per esempio, in:

```
$q2  = "SELECT DISTINCT num FROM riviste,articoli ";
$q2 .= "WHERE riviste.id=articoli.id_rivista AND ";
$q2 .= "riviste.nome='$_POST[nome]' ";
...
```

Ricordarsi di inserire, come nell'esempio, alla fine di ogni singola parte della stringa, uno spazio perché le parti sono accodate immediatamente dopo quella esistente e, se non c'è separazione, la sintassi sarebbe errata e la query non darebbe alcun risultato. L'ultima riga del frammento precedente impone la condizione che il nome della rivista sia uguale al nome che è stato selezionato nel menù a discesa. `$_POST[nome]` viene sostituito dal dato inviato e poiché si tratta di una stringa deve essere racchiuso fra apici singoli. Tutta la query è una stringa inserita in `$q` ma qui vengono utilizzati i doppi apici per delimitarla. In breve, in PHP, esistono più delimitatori di stringhe in modo che, se serve come nell'esempio, possono essere annidati uno dentro l'altro. Teoricamente i due delimitatori di stringhe potrebbero essere intercambiabili ma è importante che la stringa della query sia delimitata dai doppi apici perché questo garantisce la sostituzione della variabile (`$_POST[nome]` in questo caso) con il valore in essa contenuto.

Per poter permettere la visualizzazione della tabella risultato della query effettuata, i dati ritornati dal DB server (`$recset`) sono trasformati (`mysql_fetch_array`) in un array associativo (`$tr`) i cui elementi sono le colonne specificate nella `SELECT`.

```
<?php
  while($tr = mysql_fetch_array($recset)){
    ...
    echo $tr[num];
    ...
  };
?>
```

il comando `echo` stampa sulla pagina HTML, che verrà restituita al web server per l'invio al client che ne ha fatta richiesta, la stringa specificata e, quindi, può essere utilizzato per stampare il contenuto delle colonne della tabella risultato ma anche, per esempio, tag HTML che dovranno essere presenti nella pagina HTML finale.

La funzione `mysql_fetch_array` estrae dall'insieme `$recset` una riga della tabella. Il ciclo itera finché questa operazione risulta possibile cioè fino a che esistono righe in `$recset`.

La funzione `mysql_num_rows($recset)` restituisce la quantità di righe della tabella risultato della query `$recset` e può essere utilizzata per sapere se ci sono stati risultati.

La stampa sulla pagina HTML dei risultati di una query può essere subordinata al verificarsi di determinate condizioni:

```
if($_POST[nome]==$tr[nome])
  echo "<option selected> $tr[nome]";
else
  echo "<option> $tr[nome]";
```

Il menù a discesa della scelta del nome della rivista mostra come opzione selezionata il nome selezionato in precedenza. Anche se la pagina è ricaricata il nome scelto viene mantenuto.

```
...
if(!empty($_POST[nome]))
...
```

In questo caso il controllo verifica se, quando si è ricaricata la pagina, è stato trasmesso un valore nella variabile `nome` (il nome attribuito al menù di selezione) in modo da caricare, nel secondo menù di selezione, i numeri registrati in archivio della rivista selezionata.

```
if($_POST[nome] || $_POST[numero] || $_POST[ttit]){
  echo "<hr />";
  $q  = "SELECT titolo,pag,num,riviste.nome ";
  $q .= "FROM articoli,riviste ";
  $q .= "WHERE articoli.id_rivista=riviste.id ";

  // se c'e' il nome della rivista

  if($_POST[nome]){
    $q .= "AND riviste.nome='$_POST[nome]' ";
  };
...
```

Nell'ultimo frammento di codice, dopo aver verificato che almeno uno dei possibili input esiste, si prepara la query per la stampa della tabella dei risultati. La query, in presenza del nome della rivista (secondo controllo) dovrà essere ampliata con l'opportuna condizione.