

Corso : Tecnologie web AJAX Febbraio/Marzo/Aprile 2009

Prima Parte : Fondamenti di JAVASCRIPT

Prof. Sergio Cordella

3^ Lezione

Riferimenti web - contenuti adattati tratti da <http://web.tiscali.it/genero/javascript/>
<http://it.wikibooks.org/wiki/JavaScript/>

Le Funzioni

Le **funzioni** sono i "**mattoni**" per la costruzione di un programma in JavaScript.

Una funzione è un sottoprogramma che contiene una o più istruzioni ed esegue uno o più compiti.

Ora voi vi chiederete: **qual'è allora il programma principale?**

E' rappresentato semplicemente da tutte le istruzioni che non sono all'interno di funzioni (**in C++** il programma principale viene scritto all'interno della **funzione main()**, in **JavaScript** invece **questo non accade** e viene scritto semplicemente dopo il tag **<SCRIPT>**).

Ogni funzione ha un nome che viene utilizzato quando si vuole richiamare (e quindi utilizzare) tale funzione; **per richiamare una funzione dal programma principale** occorre scrivere il nome della funzione seguita dalla coppia di parentesi **()** per distinguerle dalle variabili;

es: **myfunction()** esegue il codice contenuto all'interno della funzione **myfunction**.

Vediamo ora come si crea una funzione:

```
<SCRIPT language="javascript">
```

```
    alert("siamo nel programma principale");
```

```
    miafunzione();           //chiamata alla funzione
```

```
function miafunzione() {  
    alert("siamo all'interno della funzione miafunzione()");  
}
```

```
</SCRIPT>
```

Occorre inserire prima del nome della funzione l'istruzione **function** che crea appunto la funzione.
Anche se all'interno di una funzione vi è solo una istruzione va sempre racchiusa tra parentesi graffe.

Perché usare le funzioni ?

- 1. Permettono di riutilizzare parte di codice poichè possono essere richiamate più volte**
- 2. Rende il listato più chiaro e leggibile**
- 3. Il codice viene eseguito solo se la funzione viene chiamata**

Passaggio parametri

Una cosa che deve essere chiara da subito riguarda il **passaggio di parametri** tra **istruzione chiamante** e **funzione**. Vediamo un esempio:

```
<SCRIPT language="javascript">
  var numero = 4;
  potenza(numero);
  //viene chiamata la funzione potenza() e
  //gli viene passato il numero
  function potenza(numero) {
    numero *= numero;
    //viene calcolato il quadrato del numero
  }
</SCRIPT>
```

La variabile numero prima della chiamata alla funzione potenza() ha valore **4** mentre **dopo ha valore 16**.

Ora vediamo come **includere codice javascript all'interno di un documento HTML** ovvero inserirlo in un URL dopo lo specificatore di pseudo-protocollo javascript:.

Questo particolare tipo di protocollo **specifica che il corpo dell'URL è codice javascript che deve essere eseguito dall'interprete javascript**.

Se l'URL contiene **più istruzioni queste devono essere separate da un punto e virgola**. Per cui è possibile per esempio, al click di un collegamento, chiamare una determinata funzione:

```
javascript:alert('ciao a tutti!!!')
```

Il codice appena scritto è un URL e come tale deve essere inserito tra i tag HTML <a> e :

```
<a href="javascript:alert('ciao a tutti!!!')">cliccami !!!</a>
```

Come passare **alla funzione scrivi()** il testo da scrivere all'interno della casella di testo .

Richiamiamo una funzione che tratta le seguenti scritte:

Ciao, come va?

Io mi chiamo Antonio

E tu come ti chiami?

Ecco l'intera pagina HTML:

```
<html>
<body>
<a href="javascript:scrivi('Ciao, come va?')">Ciao, come va?</a>
<a href="javascript:scrivi('Io mi chiamo Antonio')">Io mi chiamo Antonio</a>
<a href="javascript:scrivi('E tu come ti chiami?')">E tu come ti chiami?</a>
<form name="casella">
  <input type="text" size="20" name="T1">
</form>
<script language="javascript">
function scrivi(testo) {
  document.casella.T1.value=testo;
}
</script>
</body>
</html>
```

Variabili locali e globali

Come si può notare all'interno del corpo della funzione del listato sottostante vi sono dichiarate alcune variabili, ebbene queste variabili sono **dette locali** in quanto nascono e muoiono all'interno della funzione. Una **variabile globale** invece è quella variabile che viene dichiarata al di fuori della funzione. Infatti, una variabile locale e una globale che hanno lo stesso nome, non hanno nulla in comune.

Attenzione, è necessario la parola chiave `var` per creare le variabili all'interno delle funzioni.

```
<SCRIPT>
function potenza(bas, espon) {
    var pt = 1;
    if(espon == 0)
        return 1;
    else {
        for(y = espon; y >= 1; y--)
            pt *= bas;
        return pt;
    }
}

var x = 3, y = 5;
var k = potenza(x, y);
document.write("<br>",k);
</SCRIPT>
```

L'esempio precedente utilizza l'istruzione **return** per ritornare al programma principale il risultato della potenza.

```
var k = potenza(x, y);
```

La variabile k conterrà il risultato che poi verrà visualizzato nella pagina:

```
document.write("<br>",k);
```

Differenze tra le funzioni in JavaScript e quelle in C++

- in JavaScript non si utilizzano i prototipi;
- non occorre specificare il tipo di dato passato alla funzione per cui può essere indifferente un numero, una stringa, una data e un valore booleano...
- in C++ il programma principale viene scritto all'interno della funzione `main()` mentre in JavaScript subito dopo il tag `<script>`.

Gli script e gli eventi

Fino a questo momento abbiamo visto che gli script **vengono eseguiti in maniera sequenziale**, nell'ordine con cui sono presenti nella pagina html.

Gli script **possono essere eseguiti anche in base a degli eventi** che si possono verificare come : il **caricamento di un documento**, il **passaggio del mouse** etc

Per tale motivo tali eventi sono raggruppati in :

Eventi attivabili dai tasti del mouse

A questo gruppo si possono ricondurre i seguenti eventi:

1. **onClick**: attivato quando si clicca su un oggetto;
2. **onDblClick**: attivato con un doppio click;
3. **onMouseDown**: attivato quando si schiaccia il tasto sinistro del mouse;
4. **onMouseUp**: attivato quando si alza il tasto sinistro del mouse precedentemente schiacciato;
5. **onContextMenu**: attivato quando si clicca il tasto destro del mouse aprendo il ContextMenu.

Esempio: cliccate qui, se rispondete OK il link si attiva se rispondete Annulla il link non si attiva... ecco il codice:

```
<A HREF="link.htm" onclick="return(confirm('Sei sicuro'))">
```

Il vantaggio è che l'evento **onClick** si attiva prima del tag associato per cui se è un link, questo è caricato dopo il completamento dell'istruzione associata. In tal modo questa caratteristica si può applicare per i radio o i checkbox per non selezionarli, e per i bottoni, compresi quelli Submit e Reset, per considerarli non premuti, tranne che per un piccolo bug che rende non funzionabile l'opzione per il Reset su alcune piattaforme.

Altri eventi legati al mouse :

6. **onMouseOver**: attivato quando il mouse si muove su un oggetto;
7. **onMouseOut**: attivato quando il mouse si sposta da un oggetto;
8. **onMouseMove**: si muove il puntatore del mouse, ma poiché questo evento ricorre spesso (l'utilizzo del mouse è frequente), non è disponibile per default, ma solo abbinato con la cattura degli eventi, che si spiegherà in seguito.

Gli eventi **onMouseOver** e **onMouseOut** sono complementari in quanto il primo è attivato nel momento in cui il puntatore è posto nell'area dell'oggetto il cui tag contiene l'evento e il secondo quando ne esce.

Eventi della tastiera

- **onkeypress** si verifica quando l'utente preme un tasto sulla tastiera quando il *focus* è sull'oggetto specificato
- **onkeydown** e **onkeyup** si verificano quando l'utente schiaccia un tasto e lo rilascia. Eventi degli elementi HTML

- **onfocus** e **onblur** si verificano quando l'utente sposta il focus sull'oggetto in questione e quando toglie il focus dall'oggetto
- **onchange** si verifica quando l'utente modifica il contenuto dell'oggetto (è applicabile solo ai campi di modulo)
- **onsubmit** e **onreset** si possono applicare solo a moduli HTML e si verificano quando l'utente invia il form (pulsante di *submit*) o quando lo resetta (pulsante *reset*)
- **onselect** si verifica quando l'utente selezione del testo

Eventi della finestra

- **onload** e **onunload** si verificano quando la pagina viene caricata o quando viene chiusa
- **onresize** si verifica quando l'utente ridimensiona la finestra del *browser*
- alcuni browser supportano anche l'evento **onscroll** che si verifica allo scrolling della pagina

Eventi come attributi HTML

Il modo più semplice di definire un evento è quello di inserirlo come **attributo agli elementi HTML** della pagina stessa. Ad esempio:

```

```

Nell'esempio associamo il codice JavaScript che mostra l'alert al click dell'utente sull'immagine HTML.

Esempio con lancio di funzione :

```

```

Tramite questo sistema possiamo usufruire dell'oggetto **this** per riferirci all'elemento HTML a cui è stato applicato. Ad esempio:

```

```

Con questa riga di HTML e qualche istruzione JavaScript creiamo quello che si chiama un *rollover*, cioè l'effetto di cambiamento dell'immagine quando il mouse passa sopra di essa.

Restituire dei valori

Quando si collega del codice ad un evento, è possibile impostare un **valore di ritorno**: se questo è **false**, verrà annullata l'**azione predefinita**:

```
<a href="pagina.htm" onclick="return confirm('vuoi veramente seguire il link?');">clliccami!</a>
```

Cliccando sul link verrà mostrato un messaggio di conferma; se si clicca il pulsante *annulla* la funzione restituirà **false** e di conseguenza restituirà **false** il codice collegato all'evento; in questo modo si elimina l'azione predefinita per l'evento click su un link, cioè non verrà caricata la pagina. Questa funzionalità serve soprattutto per la **convalida dei FORM** : si collega all'evento **onsubmit** del form una funzione che restituisce **false** nel caso di errori nella compilazione dei campi.