Interactive Web Design & Development

Prototipazione di pagine e interfacce web interattive con programmazione in Javascript. Utilizzo della libreria JQuery

Unità Didattica UD01: Introduzione alla programmazione lato client

prof. Giovanni Borga

La programmazione

Cos'è la programmazione

La programmazione, in informatica, è un'attività con cui possiamo far svolgere ad un computer delle azioni connesse ad un particolare obiettivo.

Non tutti i «listati» di codice sono programmi

L'HTML come abbiamo visto non è un linguaggio di programmazione in quanto non permette di far compiere delle scelte al computer.

Gli elementi di base della programmazione sono sostanzialmente 4:

- **COMANDI**
- **COSTANTI**
- **VARIABILI**
- **FUNZIONI**

I comandi

I comandi, anche detti istruzioni, sono:

UN SET DI PAROLE CHIAVE FINITO (come i tag HTML)

messo a disposizione dal tipo di linguaggio scelto per programmare.

I comandi hanno dei **nomi univoci** che non vanno mai utilizzati per altri scopi (es. per variabili o funzioni).

Per **utilizzare un comando** è sufficiente scrivere il suo nome nel punto del flusso in cui vogliamo che accada l'effetto per cui il comando è stato creato.

Molti comandi:

a) Accettano dei dati in ingresso
b) Restituiscono dati in uscita
c) Fanno entrambe le cose

Costanti e variabili

Costanti e variabili sono dei «contenitori di dati» dotati:

a) di un proprio nome
b) di un valore (ovvero di un dato)

(come gli attributi dei tag HTML)

Le **costanti** sono caratterizzate dal fatto che il valore ad esse associato non è modificabile durante l'esecuzione del programma.

Le **variabili** invece vengono espressamente utilizzate come «veicolo di trasporto» di dati e informazioni e il valore ad esse associato può essere cambiato.

Tipicamente quindi:

Per le costanti svolgeremo solo un'azione:

UTILIZZO (lettura del valore)

Per le variabili svolgeremo due azioni:

ASSEGNAZIONE (scrittura del valore)

UTILIZZO (lettura del valore)

Costanti e variabili devono inoltre essere create, ovvero DICHIARATE. La dichiarazione può avvenire in modo isolato, oppure può essere fatta contestualmente all'assegnazione del valore.

Funzioni

Le funzioni non sono altro che sequenze di comandi

Sono come dei «mini-programmi» dentro al programma principale e servono per l'appunto a svolgere più volte la stessa sequenza senza doverla riscrivere.

Per utilizzare le funzioni occorre seguire due distinte fasi:

1. **Definizione**
2. **Chiamata**

La definizione viene svolta per prima e una sola volta
serve a indicare al programma **che sequenza** eseguire

Le chiamate possono essere multiple e vengono svolte dopo la definizione
servono a indicare al programma principale **quando** eseguire la sequenza.

(come per gli stili CSS che devono essere definiti e poi applicati)

Il linguaggio Javascript

Javascript

JavaScript è un linguaggio di scripting lato-client, che viene interpretato dal browser.

Il web funziona a due livelli:

1. le pagine web vengono inviate all'utente da un web server, cioè da un programma che si trova su un computer remoto, e che per lo più non fa nient'altro che inviare le pagine a chi ne fa richiesta
2. l'utente che naviga visualizza sul proprio browser le pagine che gli vengono inviate.

Un "browser" è un programma che permette di leggere le pagine scritte in linguaggio HTML. Quando visualizziamo le pagine web ci sono dunque due computer che si parlano: il server e il client.

Alcuni linguaggi di scripting (asp, php, perl) vengono eseguiti dal web server (si chiamano appunto linguaggi server side o lato server). JavaScript, invece, viene eseguito sul nostro computer direttamente dal browser (è un linguaggio client side o lato client).

JavaScript è un linguaggio lato client. Ovvero gli script hanno validità all'interno delle singole pagine web, e non da una pagina all'altra.

Javascript

I **linguaggi di scripting** sono definiti tali perché **la sintassi degli script si può scrivere direttamente dentro la destinazione** ed eseguire **senza doverli compilare.**

Con i linguaggi di programmazione invece (come il C, il Java ecc.) la sintassi va passata ad un compilatore, che produce un file binario in cui la sintassi scompare.

I programmi di Windows ad esempio sono dei file compilati e sono riconoscibili dal formato EXE. In essi non c'è più traccia della sintassi originaria (cioè del codice "sorgente").

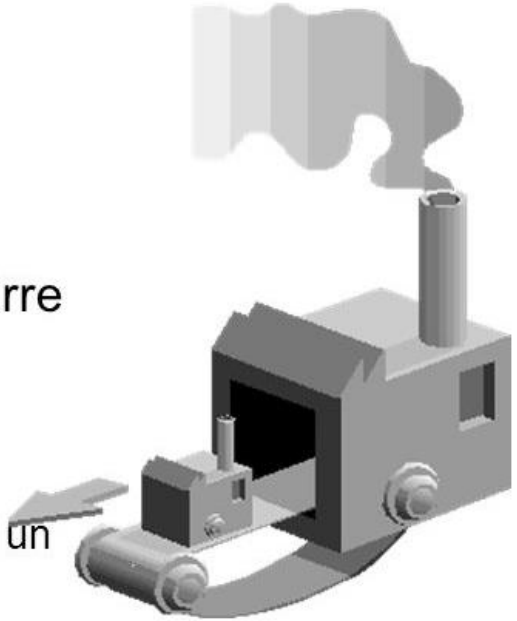
JavaScript non è compilato: è possibile quindi visualizzare in qualsiasi momento il codice di una pagina HTML e leggere le righe di codice JavaScript.

Dire che è un linguaggio di scripting sottintende anche il fatto che si tratta di un **linguaggio interpretato**: ovvero l'assenza del compilatore impone che sia lo stesso browser a «tradurre» i comandi in operazioni mediante un apposito «motore di scripting» che legge le parti di codice JavaScript.

La programmazione orientata agli oggetti (OOP): oggetti e classi

- Gli ***oggetti*** sono generati da una ***classe***
  - Si dicono anche ***istanze*** della classe

- La ***classe*** è uno schema per produrre una categoria di oggetti identici di struttura
  - La classe costituisce il *prototipo*
  - La classe descrive le caratteristiche di un oggetto
  - Una classe è una fabbrica di istanze: possiede lo schema e la tecnica di produzione



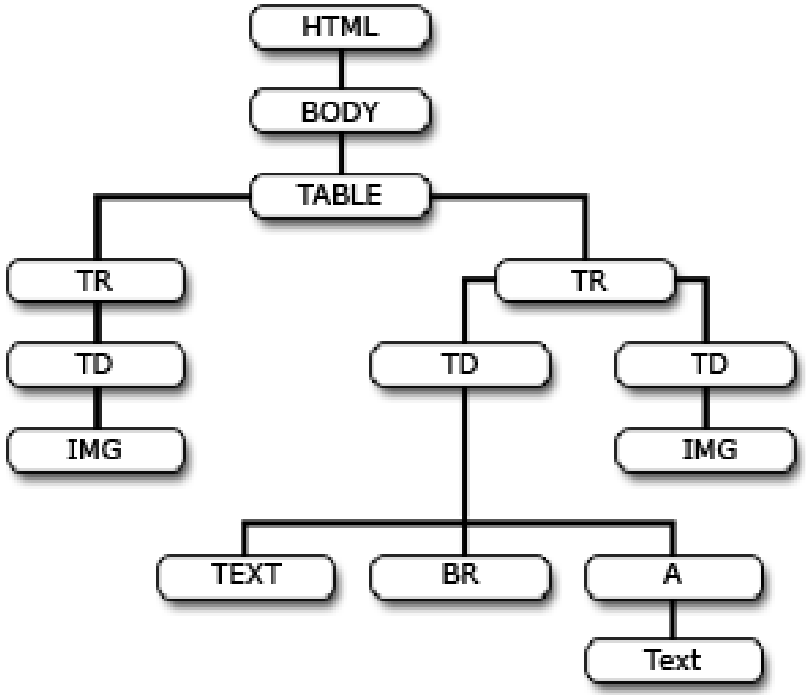
Oggetti e pagina HTML

La nostra pagina HTML è governata dal Document Object Model (DOM)

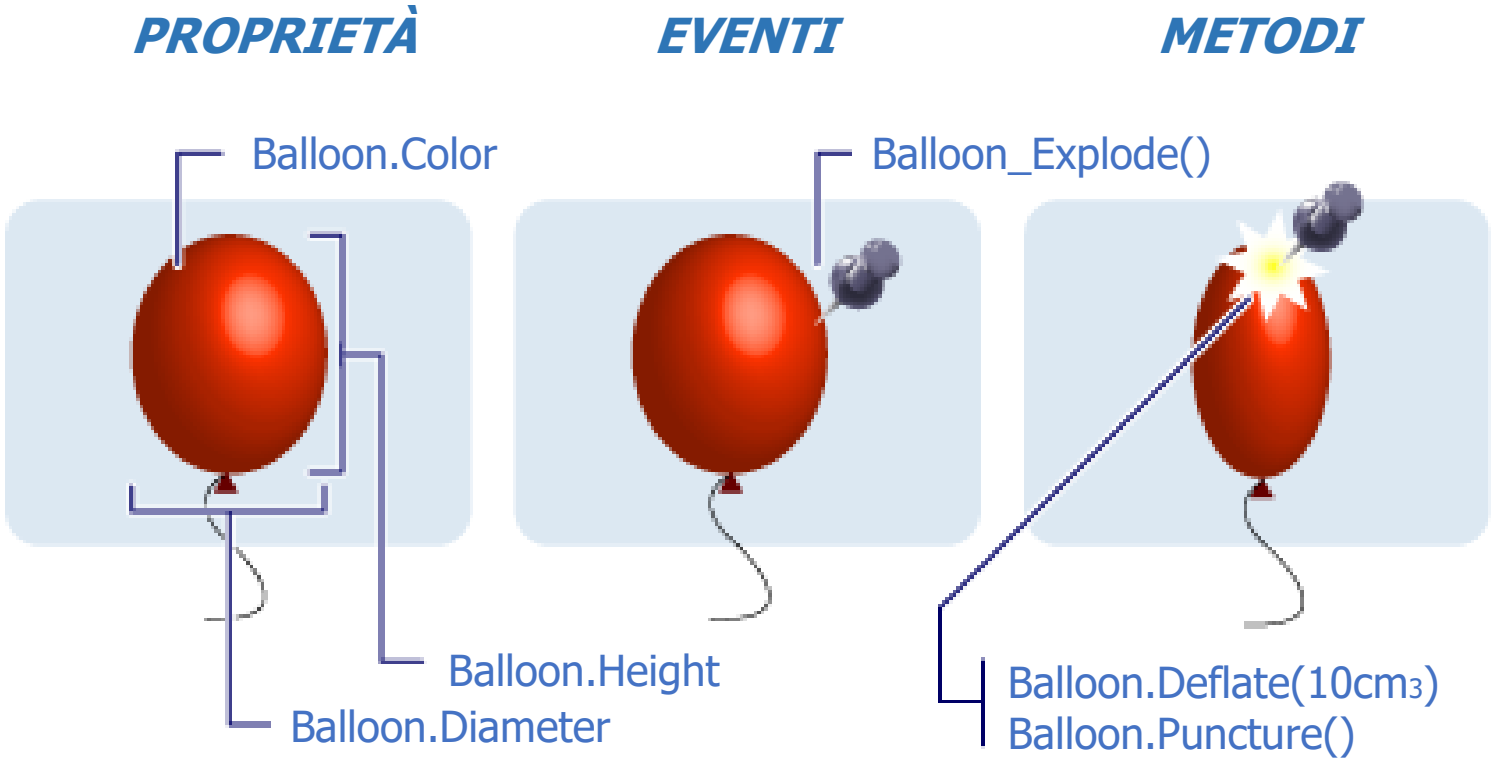
Ogni elemento HTML è un oggetto anche per la programmazione Javascript.

Tutti gli elementi HTML sono oggetti utilizzabili da Javascript principalmente tramite i rispettivi ID.

E' anche possibile far interagire gli script tramite i type selector o le classi anche se è una tecnica molto meno efficiente.



Caratteristiche degli oggetti



Utilizzo delle caratteristiche degli oggetti

PROPRIETÀ

- *Si **scrivono** (se non sono readonly)*
- *Si **leggono** (se non sono writeonly)*

METODI

- *Si **invocano** (opzionalmente inviando delle informazioni necessarie) ed eventualmente si registra l'effetto (si recupera un'informazione di ritorno)*

EVENTI

- *Si **intercettano** (e si associano ad un comando)*

Esempi di proprietà metodi ed eventi Javascript/HTML

PROPRIETÀ

- ***top***, ***bottom*** *(riferiti ad un qualsiasi elemento HTML corrispondono alle rispettive proprietà CSS)*
- ***value*** *(riferito ad una casella di testo corrisponde alla stringa contenuta nella casella)*

METODI

- ***open()*** *(invocato dall'oggetto window – il browser – apre una nuova finestra)*
- ***write()*** *(invocato dall'oggetto document – la pagina – scrive qualcosa al suo interno)*

EVENTI

- ***onclick()*** *(riferito ad esempio ad un pulsante si verifica quando esso viene premuto con il click del mouse)*

Sintassi di Javascript

Sintassi di base

Alcune informazioni di base sulla sintassi Javascript:

1. **Javascript è CASE SENSITIVE!**
2. Al termine di ogni riga di codice occorre mettere un punto e virgola (**;**)
3. I tipi di dato utilizzabili sono sostanzialmente quattro:
   - **Numerico**: si utilizzano le **cifre** senza nessun delimitatore. Il separatore decimale è il punto.
   - **Stringa**: si utilizzano tutti i **caratteri** delimitandolo con doppio o singolo apice.
   - **Booleano**: il classico valore binario. Si utilizza scrivendo ***True*** o ***False*** senza alcun altro carattere o delimitatore
   - **Nullo**: significa assenza di dato. Si utilizza scrivendo ***Null*** senza alcun altro carattere o delimitatore.
3. I commenti:

```
// commento di una sola riga
/* commento su più righe */
```

Sintassi di base

4. Per assegnare un valore ad una variabile si usa il simbolo di uguale (**=**)

5. Quando il simbolo di uguale si utilizza come operatore di comparazione in una espressione si deve scrivere due volte. Es:

```
if (x == 5) { .... }
```

6. Le **parentesi tonde** () si utilizzano per lo più per racchiudere gli argomenti delle funzioni mentre quelle graffe il codice da eseguire. Es:

```
function saluta (nome) { alert(nome) } ;
```

7. Le **parentesi graffe** {} si usano anche per i blocchi di codice del costrutto IF – ELSE - END IF

8. Le **parentesi quadre** [] si usano per gli indici di vettori e matrici. Es:

```
alunni [4]
```

Lavorare con le variabili

Creare (dichiarare) una variabile:

```
var mioNome;
```

Attribuire il valore ad una variabile:

```
mioNome = "pippo";
var mioNome = "pippo";
```
(in questo caso faccio contemporaneamente la dichiarazione)

Utilizzare una variabile (ad esempio usarne il valore per modificare il titolo della pagina):

```
document.title = mioNome;
```

Posizione del codice Javascript

Il codice Javascript può essere scritto in qualunque punto all'interno del tag HEAD o del tag BODY.

La cosa fondamentale è che sia contenuto nel suo tag <script>

```
<script>
    alert("ciao");
</script>
```

NB: è opportuno che

- ***nel tag HEAD vengano inserite le definizioni di funzioni e le inclusioni di librerie esterne***
- ***nel tag BODY tutti gli altri script***

Inclusione di script esterni

La logica di inclusione degli script è esattamente la stessa di quella dei fogni di stile.

Come i CSS i file javascript sono semplici file di testo salvati con una propria estensione: **.js**

All'interno di un file js deve essere scritto **solo codice con sintassi Javascript.**

Per includere un file javascript si inserisce dentro all'HEAD un tag **<script>** con la seguente sintassi:

```
<script src="scripts/mioscript.js"></script>
```

Con questa tecnica è possibile **utilizzare librerie anche molto complesse** dentro alle nostre pagine web avendo a disposizione numerosissime funzioni pronte per gli scopi più diversi.

Molte librerie javascript servono per creare effetti ed animazioni e consentono di migliorare sensibilmente l'usabilità e l'esperienza utente delle pagine web.

L'importante è che queste librerie siano costantemente aggiornate e ben documentate.

Utilizzo delle funzioni in Javascript

Le funzioni sono una specie di «contenitore» in cui racchiudere del codice. Anziché complicare la pagina web mischiando codice HTML e linguaggio JavaScript, è sufficiente **inserire il codice in una funzione e richiamare la funzione** quando occorre.

La sintassi necessaria per creare una funzione è questa:

```
function nomeFunzione() {
    // codice da eseguire ...
}
```

Le definizioni devono essere contenute in un tag SCRIPT posizionato nell'HEAD del documento.

Il codice da eseguire deve essere contenuto all'interno delle parentesi graffe, che delimitano l'inizio e la chiusura della funzione.

La funzione definita come sopra potrà essere richiamata semplicemente digitando:

```
nomeFunzione();
```

Esempio di funzione

```
<!DOCTYPE html>
<html>
     <head>
          <meta charset="UTF-8">
          <title></title>
          <script type="text/javascript">
               function avvisa() {
                    alert('Hai cliccato sul link');
               }
          </script>
     </head>

     <body>
          <div>
               <a href="#" onclick="avvisa();">Click</a>
          </div>
     </body>
</html>
```

Funzioni con argomenti

In molti casi può essere utile **trasferire dei dati a una funzione** in modo che questa li riceva e li utilizzi all'interno del proprio codice.

In questo modo non siamo costretti a scrivere una funzione quando cambiano solo alcuni dati mentre il resto del codice rimane lo stesso: basta scrivere il codice una sola volta e **individuare i dati che è opportuno trasformare in «argomenti» della funzione**, ovvero delle variabili «di trasporto» dei dati tra l'esterno e l'interno di essa.

Ad esempio la funzione vista precedentemente può avere quest'altra forma:

```
function avvisa(messaggio) {
    alert(messaggio);
}
```

… ed essere richiamata in questi modi:

```
<a href="#" onclick="avvisa('hai cliccato sul primo link');">Click 1</a>

<a href="#" onclick="avvisa('hai cliccato sul secondo link');">Click 2</a>
```

Funzioni con risultati

Quando definiamo una variabile con var dentro una funzione, questa avrà validità solamente all'interno di questa; in pratica **è come se le funzioni fossero delle scatole chiuse**, e tutto quello che succede all'interno di una funzione non ha nessuna validità al di fuori. Ma **come fare per comunicare con l'esterno**?

Esiste l'istruzione **return** che permette di trasferire un dato all'esterno della funzione.

L'applicazione è la seguente:

```
function nome_ateneo() {
    return('Università Iuav di Venezia');
}
```

In questo modo la chiamata `nome_ateneo()` può essere utilizzata in sostituzione del dato restituito dalla funzione. Ad esempio in questo modo:

```
<a href="#" onclick="alert(nome_ateneo());">Come si chiama l'ateneo?</a>
```

Controlli condizionali

I controlli condizionali permettono di valutare situazioni ed eseguire cose diverse in funzione dell'esito della valutazione. Il controllo condizionale classico nella programmazione è **IF**.

La sua sintassi è la seguente:

```
if (espressione da verificare) {
   //istruzioni
}
```

In questo esempio il messaggio comparirà solo se a X asseginamo il valore 5:

```
x = 5; // sostituire x con un valore a piacere
if (x == 5) {
   alert("la variabile x è uguale a 5");
}
```

Controlli condizionale IF nella forma IF – ELSEIF - ELSE

IF può essere utilizzato per effettuare valutazioni consecutive come in questo esempio:

```
if (nome == "Mario") {
   alert("ciao Mario");
}
else if (nome == "Gianni") {
   alert("ciao Gianni");
}
else {
   alert("non ti conosco");
}
```

Controlli iterativi

Oltre ai controlli condizionali, con la programmazione è tipicamente possibile svolgere un secondo tipo di operazione: le iterazioni, ovvero la ripetizione di un insieme di azioni per un numero *n* di volte.

L'interruzione dei cicli solitamente avviene per mezzo di un controllo condizionale o di un evento definito.

I controlli iterativi in javascript sono 3: **WHILE**, **DO** e **FOR**.

Per quanto riguarda il primo, l'esempio che segue inserirà nel corpo del documento 10 righe con un numero finale crescente:

```
var i = 0;

while (i < 11) {
    document.write("<p>Ciclo n." + i + "</p>"); // operazioni da ripetere
    i++; //aumento l'indice di 1
}
```

Controlli iterativi

Il controllo DO va usato in combinazione con WHILE; l'esempio che segue produce lo stesso effetto del precedente:

```
var i = 0

do {
    document.write("<p>Ciclo n." + i + "</p>"); // operazioni da ripetere
    i++; //aumento l'indice di 1
} while (i < 11);
```

NB: a differenza del primo esempio, l'operazione viene svolta comunque almeno una volta, indipendentemente dal valore di i.

Controlli iterativi

Il ciclo FOR ha infine questa sintassi di applicazione:

```
for(<inizializzazione_indice> ; <condizione_da_valutare> ; <incremento_indice>) {//istruzioni}
```

Ecco l'esempio che produce l'effetto dei due precedenti:

```
for (i = 0; i < 11; i++) {
    document.write("<p>Ciclo n." + i + "</p>"); // operazioni da ripetere
};
```

NB: si noti come non sia necessario né dichiarare i, ne incrementarlo all'interno del ciclo in quanto è tutto indicato tra le parentesi tonde.

Alcuni comandi per iniziare

Come per tutti i linguaggi di programmazione, per tutte le operazioni di uso più frequente esistono apposite funzioni che non serve definire ma che sono parte del **set di comandi standard**.

Vediamo alcuni dei comandi più frequentemente utilizzati:

```
alert('Ciao mondo!'); // avviso pop-up con interruzione del flusso
var risposta = confirm('Vuoi proseguire?'); // pop-up con interruzione e scelta
document.write('<p>Ciao mondo!</p>'); // scrittura di contenuti nel body

var titolo = document.getElementById('titolo'); // accesso ad un tag tramite il suo ID
titolo.style.color = '#ff0000'; // accesso alle proprietà css di un tag
```

Gli altri comandi, nonché le proprietà e i metodi degli oggetti sono così numerosi che non ha senso descriverli ad uno ad uno. Compresa la sintassi di base è più opportuno rifarsi alla ricca documentazione online, ad iniziare da:

https://www.w3schools.com/js/