Introduzione a Javascript

Linguaggio di scripting non di programmazione: può controllare il comportamento del browser e il suo contenuto (*mettendo a disposizione costrutti per implementare algoritmi strutturati*) ma non può effettuare elaborazioni grafiche, realizzare animazioni vere e proprie o gestire la rete per ragioni di sicurezza:

- può controllare quasi totalmente l'aspetto dei documenti HTML
- può controllare il comportamento del browser e il suo contenuto
- gestisce efficacemente i moduli per mezzo dell'oggetto Form
- gestisce **azioni** collegate ad eventi generati dagli utenti
- crea e gestisce i *cookies* (piccoli file di testo ASCII memorizzati lato client: tecnologia che immagazzina informazioni nel computer del visitatore, e rimanda le stesse informazioni automaticamente quando il visitatore torna sulla stessa pagina)
- non può controllare la macchina dell'utente
- non può leggere o scrivere file
- non può controllare periferici di I/O

Le caratteristiche[1] peculiari di **Javascript** sono:

- Linguaggio ***interpretato***;
- Linguaggio basato sugli oggetti: ***object-based*** cioè si possono usare *proprietà* (attributi e metodi) di oggetti predefiniti e crearne di nuovi (non "orientato" agli oggetti: non si può usare ad esempio il concetto di ereditarietà e polimorfismo[2]; JavaScript, infatti, non supporta le classi)
- Linguaggio guidato dagli eventi(***event-driven***)
- Il nucleo del linguaggio è incorporato nel browser
- Esiste in almeno due varianti: ***lato client*** (più comune per ***rendere dinamiche le pagine Web*** cioè inserisce contenuti eseguibili: programmi che interagiscono con l'utente, controllano browser e creano dinamicamente contenuti HTML nuovi) o *lato server*.

[1] da P.Gallo, F.Salerno " HTML, CSS, Javascript" Minerva Italica
[2] Per un'introduzione al confronto con un linguaggio Orientato agli Oggetti (Java)

Cosa c'è in Javascript

Gli elementi[3] di un programma Javascript possono essere divisi in 5 categorie:

- variabili e valori
- espressioni ed operatori
  - per manipolare i valori
- strutture di controllo
  - per modificare l'ordine di esecuzione
- funzioni
  - per raggruppare blocchi di istruzioni
- oggetti ed array (con dimensione dinamica, anche non omogenei)
  - raggruppamenti di dati e funzioni

Commenti in Javascript

Stessa tecnica del C++ o di Java

Commenti su di una singola linea //
Commenti su più linee Inizio commento /*
Fine commento */

Punto e virgola opzionale

Ogni istruzione dovrebbe terminare con il punto e virgola (;)
Può essere omesso se due istruzioni sono separate da un invio a = 3
b = 4
Nel seguente caso è obbligatorio a = 3; b = 4;
Conviene sempre metterlo

Identificatori

Un identificatore è semplicemente un **nome**

In JavaScript gli identificatori si usano per dare un nome a

- Variabili
- Funzioni
- Per fornire delle etichette per alcuni cicli
- I nomi degli identificatori devono seguire delle regole (le vedremo quando tratteremo le variabili)

[3] da Carlo Blundo "Tecnologie di Sviluppo per il WEB"

Tipo delle variabili

Javascript *non è un linguaggio tipato*

- Alla stessa variabile possiamo assegnare valori di tipo differente senza generare errori

Esempio

```
<script type=" text/javascript" >
<!--
var altezza=2+3.4;
document.write(altezza);    /* funzione predefinita o metodo write per scrivere sul documento
                               con document oggetto browser dipendente o oggetto Navigator */
document.write("<BR>");   /* funzione predefinita anche metodo writeln per scrivere nel
                               browser la stringa di testo passata come parametro con “a capo” */
altezza="5.4 metri";
document.write(altezza);
// -->
</script>
```

In realtà Javascript utilizza un **controllo di tipo lasco** per cui non esiste una sezione di dichiarazione del tipo delle variabili e non c'è bisogno di farlo, ma **automaticamente viene inizialmente assegnato il tipo in base alla dichiarazione**. Ad esempio altezza="*testo*" e altezza=5.4 sono due dichiarazioni pienamente valide: nel primo caso è considerata oggetto string, nel secondo la stessa è considerata come valore numerico.

Dichiarazione di variabili

Non è necessario dichiarare le variabili anche se è *buona norma* farlo sempre

Per dichiarare una variabile possiamo semplicemente assegnarle un valore

- i =10; // a questo punto i è dichiarata
- str = "Ciao \n mondo!"; // a questo punto str è dichiarata

Per dichiarare una variabile con *buono stile* si usa la parola chiave ***var***

- var i;
- var r, d;
- var s="ciao";

Possiamo dichiarare con **var** una variabile più di una volta senza causare errori
Se si crea all’interno di una funzione una variabile senza far ricorso a var, allora si crea una variabile *globale*

Inizializzazione di variabili

Possiamo inizializzare una variabile quando la dichiariamo

- var i=10;

il manualetto completo all'URL: http://stclassi.altervista.org/Dispense/Linguaggio javaScript.pdf

Oggetti in Javascript

Un oggetto in JavaScript è una collezione[4] di valori con nome (*named values*). JavaScript **non supporta** le **classi** e, a differenza di linguaggi OO, usa oggetti di cui può modificare *proprietà* o crea tale insieme di dati (attributi) e di funzioni (metodi) con riferimento ad un prototipo cioè la funzione **costruttore**.

Per far riferimento ad una proprietà di un oggetto si fa riferimento al nome dell'oggetto seguito dal punto (.) e dal nome della proprietà. Con sintassi: *nomeOggetto.nomeAttributo*

Se la definizione del tipo dell'oggetto contiene una funzione, allora la funzione prende il nome di **metodo** dell'oggetto ed il *named value* diventa il nome del metodo. Per richiedere l'esecuzione di tale funzione si userà sempre la *dot-notation* con sintassi *nomeOggetto.nomeMetodo(par1, par2,..)*

Una variabile è fondamentalmente la stessa cosa di una *proprietà* di un oggetto

Le principali famiglie di oggetti in Javascript:

- ***"Riflessi" da HTML*** (creati automaticamente dal browser per ogni ***elemento*** di un documento HTML)
- ***Interni*** o integrati (non legati al browser né al documento HTML: Array, String, Date, Math)
- ***Browser-dipendenti***[5] (detti oggetti Navigator, "riflessi" dall'ambiente del browser)
- ***Definiti dall'utente*** (richiedono prima la definizione di una ***classe*** cioè una famiglia di oggetti con le stesse caratteristiche o **attributi** e comportamenti o **metodi** tra i quali il **costruttore** che servirà ad inizializzare i valori degli attributi poi la creazione con operatore **new**)

Esempi d'uso di oggetti raggruppati per categoria

Manipolazione del DOM
Oggetto Window
Oggetto Document
Oggetto Form
Oggetto Frame

Gestione del browser
Oggetto Navigator
Oggetto Screen

Oggetti builtin di Javascript
Oggetto Array
Oggetto Data
Oggetto Math
Oggetto Stringa

[4] Una specie di struttura in C; in genere si fa riferimento ai named values con il nome di *proprietà*
[5] Una pagina web viene scomposta da JavaScript in un modello ad oggetti (ognuno con le sue *proprietà*) in relazione reciproca. Oggetti "di livello massimo" (Navigator) nella gerarchia degli oggetti JavaScript: Window, document, location, navigator e history

Cenni su oggetti definiti dall'utente

Vediamo un esempio di **definizione** di **prototipo** cioè tipo di un oggetto in JavaScript che avviene tramite l'implementazione di quello che viene chiamato metodo '**costruttore**'.

Un costruttore, come nei linguaggi OO, non è altro che un metodo che viene richiamato quando l'oggetto viene *istanziato* e ha il compito di inizializzare la nuova istanza dell'oggetto secondo valori prestabiliti o in base ai parametri ricevuti.

Proviamo a fare una similitudine, così come la matrice rappresenta l'originale da cui stampare le banconote, così il tipo dell'oggetto, consistente nell'elenco di una serie di proprietà e funzioni che ne individuano le caratteristiche ed il comportamento è l'originale da cui, attraverso l'atto dell'istanza generare una variabile avente le sue stesse caratteristiche.

Come esempio, proviamo a definire un tipo di oggetto che possa poi essere utilizzato come un'anagrafica e cioè che sia caratterizzato da un nome, un giorno di nascita, un mese, ed un anno. Si vuole inoltre poter visualizzare nella finestra del browser, all'interno del documento, i dati di quella particolare istanza.

Procediamo quindi a scrivere il **costruttore** del tipo Anagrafica:

```
function Anagrafica (Nome, Giorno, Mese, Anno) {
    this.Nome = Nome;
    this.Giorno = Giorno;
    this.Mese = Mese;
    this.Anno = Anno;
        this.Visualizza = StampaOggetto;
        }
```

La parola chiave ***this***, utilizzata all'interno del costruttore è un modo per far riferimento ad una *proprietà* dell'oggetto sul quale si lavora.

Nell'esempio proposto sarà necessario aver precedentemente definito la funzione **StampaOggetto()**

```
<SCRIPT LANGUAGE="javascript">
<!—
        function StampaOggetto() {
            document.write ("Dati in archivio:");
            document.write ("Nome: "+this.Nome);
            document.write ("Giorno: "+this.Giorno);
            document.write ("Mese: "+this.Mese);
            document.write ("Anno: "+this.Anno);
        }

        function Anagrafica (Nome, Giorno, Mese, Anno) {
            this.Nome = Nome;
            this.Giorno = Giorno;
            this.Mese = Mese;
            this.Anno = Anno;
            this.Visualizza = StampaOggetto;
        }
//-->
</SCRIPT>
```

In questo modo, per visualizzare le *proprietà* di un oggetto di tipo Anagrafica, si **istanzia** ad esempio con il nome Antonio,

Antonio = **new Anagrafica** ("Antonio", 11, "Novembre", 1968);

e si chiederà come *servizio* all'oggetto Antonio di visualizzarle:

Antonio.**Visualizza();** // esecuzione del metodo

Codice completo della pagina web:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
<!--
function StampaOggetto ()
{
document.write ("Dati in archivio:");
document.write ("<BR>Nome: "+this.Nome);
document.write ("<BR>Giorno: "+this.Giorno);
document.write ("<BR>Mese: "+this.Mese);
document.write ("<BR>Anno: "+this.Anno);
}

function Anagrafica (Nome, Giorno, Mese, Anno) {
this.Nome = Nome;
this.Giorno = Giorno;
this.Mese = Mese;
this.Anno = Anno;
this.Visualizza = StampaOggetto;
}
Antonio = new Anagrafica ("Antonio", 11, "Novembre", 1968);   // crea istanza

Antonio.Visualizza(); // esecuzione del metodo
//-->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

In JavaScript si può **aggiungere una proprietà ad una particolare istanza** di un oggetto. Ad esempio scrivendo:

```
Antonio.Indirizzo = "Via Parma, Chiavari";
```

si aggiunge la proprietà Indirizzo all'istanza Antonio dell'oggetto Anagrafica.

Questa operazione non avrà alcun effetto sugli altri oggetti dello stesso tipo che sono già stati istanziati o che lo saranno in seguito.

Per **aggiungere una proprietà ad un oggetto** si può utilizzare la parola chiave **prototype**. In questo caso la proprietà verrà aggiunta alla definizione del tipo dell'oggetto e quindi avrà effetto su tutti gli oggetti già istanziati o che istanzieremo. La sintassi è la seguente:

```
Anagrafica.prototype.NomeProprietà = ValoreProprietà;
```

potendo inizializzare ad un valore nullo (uso della parola chiave null) come nel seguente esempio:

```
Anagrafica.prototype.Indirizzo = null;
Antonio.Indirizzo = "Via Parma, Chiavari;
Claudio.Indirizzo = "Via Merano, Mesagne";
```

Entrambe le possibilità illustrate per aggiungere proprietà **non sono un modo consigliato di procedere**; si dovrebbe infatti progettare con attenzione il tipo di oggetto e non modificarne in seguito la definizione.

Per approfondire: http://lia.deis.unibo.it/corsi/2004-2005/LABTEC-LA-CE/slides/4bis-Elementi%20di%20JavaScript.pdf