Seminario JavaScript: Programmazione orientata agli oggetti

Linguaggi per il Web

Ingegneria Informatica, Ingegneria dell'Informazione,
Sapienza Università di Roma, sede di Latina

26 novembre 2012

Dott. Corrado Di Benedetto

Argomenti

- Oggetti
- Oggetti personalizzati
- Proprietà
- Metodi
- Incapsulamento
- Ereditarietà
- Prototipi
- Panoramica oggetti nativi

Programmazione orientata agli oggetti

- La **programmazione orientata agli oggetti**

  (**OOP**, **O**bject **O**riented **P**rogramming)

  è un paradigma di programmazione che permette di definire **oggetti** software in grado di interagire attraverso lo scambio di messaggi.

Oggetti

- Gli **oggetti** sono strutture dati, costituiti da **proprietà** e **metodi,** usate per modellare le entità del dominio applicativo di interesse.
- In molti linguaggi OOP (Java, C++, C#, ecc.) la definizione di tali strutture dati è fatta utilizzando il concetto di *Classe*.
- In JavaScript non è così, la parola *class* è considera riservata, le funzioni sono utilizzate per definire la struttura di uno oggetto.

Oggetti personalizzati

- Per creare un oggetto è necessario utilizzare una funzione chiamata costruttore:

```
var variabile = new nomeOggetto ();
```

- La `variabile` sarà l'unico modo per fare riferimento all'istanza di `nomeOggetto`.

- I costruttori possono avere anche una serie di `parametri` per impostare alcune proprietà dell'oggetto creato:

```
var variabile  = new nomeOggetto (parametri);
```

Esempio

- Creazione di un oggetto di tipo `Moto`.

```
var miaMoto = new Moto();
```

- Per inizializzare la variabile `miaMoto` dobbiamo usare la funzione costruttore del prototipo moto che prevede due parametri `modello` e `colore`:

```
var miaMoto = new Moto("Honda", "Rosso");
```

Proprietà

- Gli oggetti possiedono delle caratteristiche dette **proprietà,** ad esempio una moto ha come proprietà: la marca, il modello, la cilindrata, le dimensioni, il costo, il colore, ecc.

- Il valore di una proprietà viene attribuito inizialmente dal costruttore e successivamente viene modificato agendo sull'oggetto.

- Per fare riferimento alla proprietà si usa la sintassi:

```
nomeOggetto.nomeProprietà;
```

Proprietà

- Una proprietà può essere in sola lettura.
- Alcune proprietà sono legate a caratteristiche intrinseche dell'oggetto, altre sono determinate dal prototipo in base a dati forniti in precedenza.
- Per cambiare la proprietà `colore`:

```
miaMoto.colore = "rosso";
```

Metodi

- Gli oggetti hanno la possibilità di effettuare delle operazioni: **metodi.**

- Ad esempio una moto ha come metodi: metterla in moto, guidare, frenare, ecc.

- Queste operazioni potranno modificare alcune delle sue proprietà come il livello del suo carburante o la velocità.

Ereditarietà

- Il meccanismo dell'ereditarietà permette di derivare nuovi **tipi**, detti **sottotipi**, a partire da quelli già definiti **prototipi** (relazioni **is-a**). Un sottotipo mantiene i metodi e gli attributi delle prototipo da cui deriva; inoltre, può definire i propri metodi o attributi e può ridefinire il codice di alcuni dei metodi ereditati (overriding).

- E’ utilizzato per ottenere l'estensibilità e il riuso del codice.

Prototipi

- Per creare un nuovo oggetto è possibile partire da un **modello** che in JavaScript è un oggetto chiamato **prototipo.**

- Il prototipo indica come creare oggetti dello stesso tipo e quindi ogni oggetto è un'istanza del prototipo.

Oggetti Nativi

Gli oggetti nativi (built-in) che passeremo in rassegna sono:

- Global
- String
- RegExp
- Array
- Date
- Math
- Boolean
- Number
- Function

Standard ECMA-262 v. 5.15.1 Edition, June 2011, Standard Built-in ECMAScript Objects, p.102

Oggetto Global

- L'oggetto Global permette di raggruppare alcune proprietà e metodi a livello più alto, non hanno un oggetto padre.

- L’oggetto Global viene creato prima che si acceda al contesto di esecuzione, non ha costruttore, non è possibile invocarlo come funzione.

Standard ECMA-262 v. 5.15.1 Edition, June 2011, The Global Object, p.103

Oggetto String

- L'oggetto `String` permette di effettuare numerose operazioni sulle stringhe, come ricerca, isolamento di un carattere e altro ancora.

- Per creare un nuovo oggetto della classe `String` usiamo la sintassi:

```
var oString = new String(stringa);
```

- Il costruttore prende come parametro la stringa che sarà manipolata nel corso dello script

Esempi oggetto String

Proprietà:

```
var testo = "Sono una stringa"
testo.length;     // restituisce  16
```

Metodi:

```
var t1 = new String ("Roma");
var t2 = t1.toLowerCase()       //restituisce "roma"
var t3 = t1.toUpperCase()       //restituisce "ROMA"
```

Standard ECMA-262 v. 5.15.1 Edition, June 2011, String Objects, p.141

Oggetto RegExp

- L'oggetto `RegExp` permette di gestire le regular expressions (espressioni regolari); nella pratica, un'istanza della classe `RegExp` è un potente strumento per fare pattern-matching, cioè il controllo di un pattern su oggetti `String`.

- Il pattern è definito utilizzando le espressioni regolari, cioè simboli e sequenze di caratteri di controllo.

- Un esempio classico è il controllo sintattico di una e-mail, esso è composta da tre parti: username, @ e dominio del provider.

Creazione oggetto RegExp

- `var oRegExp = new RegExp (pattern, modifiers);`

oppure

- `var oRegExp = /pattern /modifiers;`

Dove pattern specifica il pattern di una espressione e modifiers specifica se una ricerca deve essere globale (`g`), case-insensitive (`i`) e multilinea (`m`).

Esempio oggetto RegExp

```
function demoShowMatchClick() {
  var re = new RegExp(document.demoMatch.regex.value);
  var m = re.exec(document.demoMatch.subject.value);
  if (m == null) {
    alert("No match");
  } else {
    var s = "Match at position " + m.index + ":\n";
    for (i = 0; i < m.length; i++) {
      s = s + m[i] + "\n";
    }
    alert(s);
  }
}
```

Oggetto Array

- L'oggetto `Array` è un tipo di dato strutturato che permette di organizzare i dati secondo un indice;

- In altre parole, lo stesso oggetto contiene a sua volta numerosi valori, ciascuno dei quali contrassegnato da una chiave numerica.

Creazione oggetto Array

- `var corsi = new Array (); // Array vuoto`

- ```
  var corsi = new Array(3);
      // Array vuoto con lunghezza 3
      corsi[0]="Basi di dati;
      corsi[1]="Matematica1";
      corsi[2]="Fisica1";
  ```

- ```
  var corsi = new Array("Basi di dati ",
              "Matematica1","Fisica1");
  ```

- ```
  var corsi = ["Basi di dati ",
              "Matematica1","Fisica1"];
  ```
  ```

Esempi oggetto Array

Proprietà

```
corsi.length;    // restituisce la lunghezza di corsi
```

Metodi

```
corsi.push("Fisica2");  // inserisce un elemento
                        // in corsi
corsi.pop();      // elimina l'ultimo elemento e
                  // restituisce il valore eliminato
corsi.shift();    // elimina il primo elemento e
                  // restituisce il valore eliminato
corsi.sort();     // ordina secondo l'ordine
                  // alfabetico
corsi.reverse(); // inverte l'ordine degli elementi
```

Standard ECMA-262 v. 5.15.1 Edition, June 2011, Array Objects, p.122

Oggetto Date

- L'oggetto `Date` permette di gestire in modo semplice e veloce le date; nella pratica, un'istanza della classe `Date` identifica una data e le funzionalità della classe permettono di aggiungere o sottrarre anni, mesi, giorni o recuperarne il valore.

Creazione oggetto Date

```
1.  new Date();                 // contente la data e
                                // l'ora corrente

2.  new Date(millisecondi);// millisecondi dal
                           // 1970/01/01

3.  new Date(dateString);   // JavaScript è eseguito
                            // con le impostazioni
                            // del client

4.  new Date(anno, mese, giorno, [ora], [minuti],
    [secondi], [millisecondi]);
                          // gennaio è il mese 0 e non 1
```

Esempi oggetto Date

```
var oDate= new Date("10-7-2007");
// Attenzione:
// per un utente USA invece che il 10 luglio 2007
// la data sarà 7 ottobre 2007.

var oDate = new Date(2007, 7, 10); // non ambiguo

var giornoS = oDate.getDay();
var giornoM = oDate.getDate();
var mese = oDate.getMonth();
var anno = oDate.getFullYear();
var nuovadata= oDate.setDate(data.getDate()+5);
```

Standard ECMA-262 v. 5.15.1 Edition, June 2011, Date Objects, p.166

Oggetto Math

- L'oggetto `Math` mette a disposizione numerose funzioni e costanti matematiche.

- E’ un oggetto particolare, non richiede di lavorare su istanze di oggetti, ma permette di accedere ai suoi metodi usando la notazione:

- `Math.nomeMetodo();`
- `Math.nomeProprietà.`

Esempi oggetto Math

Proprietà:

```
Math.E;             // restituisce il valore
                    // della costante matematica E
Math.PI;            // restituisce il valore
                    // approssimato Pi greco
Math.LN2;           //restituisce valore del
                    // logaritmo naturale di 2
```

Metodi:

```
Math.abs(x);          // restituisce il valore
                      // assoluto (o modulo) di x
Math.sqrt(x) ;        // restituisce la radice
                      // quadrata di x
Math.pow(b, n);       // restituisce l'enesima
                      // potenza del numero b
```

Standard ECMA-262 v. 5.15.1 Edition, June 2011, The Math Object, p.159

Oggetto Boolean

- L'oggetto `Boolean` permette di convertire in booleano un valore non booleano.

- Per creare un nuovo oggetto della classe `Boolean` usiamo la sintassi:

```
var oBoolean = new Boolean(valore);
```

L'oggetto sarà impostato su *false,* se `valore` è omesso, false, 0, null, "", negli altri casi su *true* .

Standard ECMA-262 v. 5.15.1 Edition, June 2011, Boolean Objects, p.152

Oggetto Number

- L'oggetto `Number` permette di effettuare operazioni sui numeri, è utile per accedere a valori costanti come il numero massimo e minimo rappresentabili, + infinito, – infinito e NaN.

- Per creare un nuovo oggetto della classe `Number` usiamo la sintassi:

```
var oNumber = new Number (valore);
```

- Il costruttore prende come parametro un `valore` numerico.

Standard ECMA-262 v. 5.15.1 Edition, June 2011, Number Objects, p.154

Oggetto Function

- L'oggetto `Function` permette di definire una stringa durante l'esecuzione e di compilarla come funzione.

- La sintassi per dichiarare un oggetto `Function` è la seguente:

```
var oFunction = new Function ([p1, p2, … , pn,] body)
```

Standard ECMA-262 v. 5.15.1 Edition, June 2011, Function Objects, p.117

Esercizi

Esercizio 1

Creare un oggetto persona contenente le proprietà: Nome, Cognome, Età Residenza.

Scrivere uno script che richieda all'utente dei valori per tali proprietà, definisca un'istanza di un oggetto con tali proprietà e stampi, in una tabella, il contenuto dell'oggetto.

Modificare lo script precedente in modo che, dopo aver inserito i valori per le sue proprietà, chieda all'utente il nome di una proprietà da aggiungere all'istanza dell'oggetto ed un suo valore. Lo script, alla fine, deve stampare nuovamente il contenuto dell'istanza dell'oggetto.

Esercizio 2

Creare un oggetto studente derivandolo dall'oggetto persona definito nell'esercizio precedente.

Testare l'oggetto creato inserendo la sua definizione in un documento HTML contenente uno script che chiede all'utente di immettere i valori delle proprietà dell'oggetto studente e visualizzi il contenuto dell'istanza dell'oggetto creato.

Esercizi

Esercizio 3

Definire dei prompt che richiedono le proprietà Processore, Hard Disk, Scheda Video, Ram di un computer. Creare un oggetto computer con tali proprietà. Stampare le proprietà dell'oggetto.

Esercizio 4

Creare una pagina in cui il sistema chiede all'utente il proprio nome

Il sistema verifica il nome e comunica all'utente se è un amministratore o meno. E' dato un array di nomi che hanno i diritti di amministratore.

Esercizio 5

Scrivere uno script che:

Chieda all'utente di inserire una serie di squadre di calcio, usando il pulsante annulla per terminare l'immissione;

Per ogni squadra, procedendo in ordine alfabetico, chieda il punteggio attuale (assumendo che sia compreso tra 0 e 99);

Sfrutti opportunamente i metodi sort e reverse per ordinare le squadre in base ai punteggi e visualizzare la classifica attuale.

Consigli: usare un array per memorizzare i nomi; combinare opportunamente la stringa del punteggio con quella del nome della squadra prima di usare il metodo sort.

Esercizi

Esercizio 5

Creare un pagina HTML che visualizzi un slide-show di immagini.

La pagina inizia visualizzando la prima immagine.

Lo scorrimento delle immagini avviene alla pressione di due tasti:

uno per passare all'immagine precedente, ed un altro per saltare alla successiva.

Raggiunta l'ultima immagine, la visualizzazione riprende dalla prima.

Suggerimenti:

Memorizzare in un array i nomi delle immagini da caricare.

Definire una variabile che contenga il numero dell'immagine attualmente visualizzata. Definire due funzioni Precedente() e Successivo() che incrementino (decrementino) la variabile, visualizzino l'immagine (cambiando un opportuno .src) e riprendano dall'inizio (dalla fine) quando necessario.

Associare le procedure Precedente() e Successivo() alle proprieta' onClick delle ancore che racchiudono le freccie. Ricordarsi di specificare retrun false; dopo la chiamata (onClick="Successivo(); return false;").

Risorse

JavaScript

- http://it.wikibooks.org/wiki/JavaScript/Introduzione
- http://it.wikipedia.org/wiki/JavaScript
- http://www.w3schools.com/js/default.asp
- http://www.w3schools.com/jsref/default.asp
- http://www.ecma-international.org/publications/standards/Ecma-262.htm;

DOM

- http://www.w3schools.com/htmldom/default.asp

Regular-Expressions

- http://www.regular-expressions.info;
- http://it.wikipedia.org/wiki/Espressione_regolare