Applicazioni web

Parte 3
Javascript, DOM

Alberto Ferrari

Sommario

- Javascript
  - Sintassi
- Oggetti DOM
- Esempi

Alberto Ferrari

JavaScript
Cos'è JavaScript?

- JavaScript è stato sviluppato da Netscape per aggiungere interattività alle pagine html
  - JavaScript è un linguaggio di scripting
  - JavaScript è un linguaggio interpretato (script eseguiti senza compilazione preliminare)
  - JavaScript è di solito racchiuso direttamente nelle pagine html
  - JavaScript è supportato da tutti i maggiori browser
- Java e JavaScript non sono la stessa cosa
  - Sono due linguaggi completamente diversi

JavaScript
Cosa si può fare con JavaScript?

- JavaScript è (anche) uno strumento di programmazione per html
  - Autori di pagine html non sempre sono programmatori
  - Ma sintassi js molto semplice
  - Quasi chiunque può inserire brevi "snippets" di codice nelle pagine
- Inserire testo dinamicamente
- Reagire ad eventi
- Leggere e scrivere elementi html
- Controllare dati

JavaScript
Script interni

- Il tag html *<script>* permette di inserire JavaScript in una pagina html

```
<html><body>
  <script type="text/javascript">
    document.write("Hello World!")
  </script>
</body></html>
```

- I browser più vecchi mostrano lo script come contenuto della pagina
  - Per impedirlo, spesso si usa un tag di commento html

```
<script type="text/javascript">
  <!--
  some statements
  //-->
</script>
```


JavaScript
Script esterni

- Eseguire lo stesso script su più pagine, senza copiarlo
- Script in un file di testo separato, es. *xyz.js*

```
document.write("This script is external")
```

- Poi si può richiamare lo script da qualsiasi pagina, usando l'attributo *src*

```
<html><body>
  <script src="xyz.js"></script>
</body></html>
```

JavaScript
Linee guida

- Maiuscole/minuscole
  - JavaScript è un linguaggio case-sensitive
  - Tratta queste tre parole come diverse
  - *example, Example, EXAMPLE*
- Punti e virgola
  - Le istruzioni possono terminare con un punto e virgola
  - … anche se non è obbligatorio
  - Necessario quando si inseriscono più istruzioni su una sola riga
- Spaziatura
  - JavaScript, come html, ignora spazi, tabulazioni e ritorni a capo che appaiono nelle istruzioni
  - Tuttavia riconosce gli spazi che fanno parte di una stringa

JavaScript
Variabili

- Si può creare una variabile con l'istruzione *var*:
  - `var someName = someValue`
- Ma si può creare una variabila anche senza *var*:
  - `someName = someValue`
- Quando una variabile è dichiarata all'interno di una funzione, la variabile è accessibile solo all'interno di questa
  - Quando si esce dalla funzione, la variabile è distrutta
  - *variabile locale*
  - Variabili locali con lo stesso nome, dichiarate in funzioni diverse, sono considerate variabili distinte
- Se si dichiara una variabile fuori da qualsiasi funzione, tutte le funzioni possono accedervi
  - *variabile globale*
  - La loro esistenza comincia con la dichiarazione e termina con la chiusura della pagina

JavaScript
Operatori

- Operatori aritmetici
  - +, -, *, /, %, ++, --
- Operatori di assegnamento
  - =, +=, -=, *=, /=, %=
- Operatori di confronto
  - ==, !=, >, >=, <, <=
- Operatori logici
  - &&, ||, !

Alberto Ferrari

JavaScript
Istruzioni conditionali

- *if*

```
if (condition) {
  statements1
}
```

- *if… else*

```
if (condition) {
  statements1
}
else {
  statements2
}
```

- *switch*

```
switch (expression) {
  case label1:
    statements1;
    break;
  case label2:
    statements2;
    break;
  ...
  default:
    statements;
}
```

Alberto Ferrari

JavaScript
Istruzioni di ciclo

- *for*
  - ```
    for (initialization; condition; increment) {
      statements
    }
    ```
- *do*
  - ```
    do {
      statements
    } while (condition)
    ```
- *while*
  - ```
    while (condition) {
      statements
    }
    ```


JavaScript
Funzioni

- Per creare una funzione, bisogna specificarne il nome, gli argomenti e le istruzioni
  - ```
    function total(a, b) {
      var result = a + b
      return result
    }
    ```
- Poi si può chiamare la funzione
  - ```
    sum = total(2, 3)
    ```
    ```

JavaScript
Oggetti

- Semplice paradigma ad oggetti
  - Un oggetto ha delle *proprietà* che possono essere variabili primitive o altri oggetti
  - Un oggetto ha anche delle funzioni denominate *metodi*
- Si può creare un oggetto in due passaggi:
  1. Si definisce un tipo di oggetto scrivendo una funzione *costruttore*
  - ```
    function Car(make, model, year) {
      this.make = make
      this.model = model
      this.year = year
    }
    ```
  2. Si crea una istanza dell'oggetto con *new*
  - ```
    myCar = new Car("Eagle", "Talon TSi", 1993)
    ```
- Si può poi rimuovere l'oggetto usando l'operatore *delete*
  - ```
    delete myCar
    ```

Alberto Ferrari

JavaScript
Proprietà degli oggetti

- Le proprietà e gli array sono intimamente correlati
- In effetti, sono diverse interfacce per la stessa struttura dati
  - ```
    myCar.make = "Ford"
    myCar.model = "Mustang"
    myCar.year = 1969
    ```
  - ```
    myCar["make"] = "Ford"
    myCar["model"] = "Mustang"
    myCar["year"] = 1967
    ```

Alberto Ferrari
    ```

JavaScript
Metodi degli oggetti

- Un metodo è una funzione associata ad un oggetto
- Un metodo si definisce allo stesso modo che una funzione
  - ```
    function displayCar() {
      var result = "A Beautiful " + this.year + " " + this.make
      + " " + this.model
      return result
    }
    ```
- La funzione si può poi associare ad un oggetto esistente
  - ```
    function Car(make, model, year, owner) {
      this.make = make
      this.model = model
      this.year = year
      this.owner = owner
      this.displayCar = displayCar
    }
    ```
- Infine si può invocare il metodo nel contesto di un oggetto
  - ```
    car1.displayCar()
    ```


JavaScript
Manipolazione di oggetti

- L'istruzione *for...in* itera una variabile sulle proprietà di un oggetto
  - Su ciascuna diversa proprietà, vengono eseguiti le istruzioni specificate
  - ```
    for (variable in object) { statements }
    ```
- L'istruzione *with* stabilisce un oggetto di default per un insieme di istruzioni
  - JavaScript controlla tutti i nomi non qualificati, e tenta di risolverli sulle proprietà dell'oggetto di default
  - Se un nome non qualificato non corrisponde ad una proprietà dell'oggetto di default, allora si usa una variabile locale o globale
  - ```
    with (object) { statements }
    ```
    ```

JavaScript
Oggetto String

- Un oggetto *String* è un wrapper attorno al tipo di dato primitivo *stringa*
  - ```
    s1 = "foo" //creates a string literal
    values2 = new String("foo") //creates a String object
    ```
- Si possono invocare i metodi di *String* anche su una stringa primitiva

- Proprietà
  - ***length*** – Numero di caratteri in una stringa
- Metodi
  - ***charAt*** – Restituisce il carattere ad una posizione specificata
  - ***indexOf*** – Restituisce la posizione di una specificata sotto-stringa
  - ***concat*** – Combina la stringa con altre (ne restituisce una nuova)
  - ***split*** – Divide una stringa in un array di stringhe
  - ***substring, substr*** – Restituisce una specifica porzione della stringa; occorre indicare l'inizio e la fine, oppure l'inizio e la lunghezza
  - ***toLowerCase, toUpperCase*** – Restituisce una stringa tutta in minuscolo o maiuscolo

JavaScript
Oggetto Array

- Un oggetto *Array* è usato per memorizzare un insieme di valori in una singola variabile
  - Ogni valore è un elemento dell'array ed è associato ad un indice numerico
  - ```
    var family_names = new Array(3)
    family_names[0] = "Tove"
    family_names[1] = "Jani"
    family_names[2] = "Stale"
    ```

- Proprietà
  - ***length*** – Numero di elementi nell'array
- Metodi
  - ***concat*** – Concatena l'array con altri (ne restituisce uno nuovo)
  - ***join*** – Restituisce la concatenazione di tutti gli elementi, come stringa
  - ***reverse*** – Rovescia l'array
  - ***slice*** – Restituisce una specifica porzione dell'array
  - ***sort*** – Ordina l'array

JavaScript
Oggetto Date

- Un oggetto *Date* serve per memorizzare istanti di tempo
- Gran numero di metodi, ma nessuna proprietà pubblica
  - ***getFullYear(), getMonth(), getDate(), getHours(), getMinutes(), getSeconds(), getMilliseconds()*** – Restituiscono anno (4 cifre), mese (0-11), data (1-31), ora (0-23), minuti (0-59), secondi (0,59), millisecondi (0-999) di un oggetto *Date*
  - ***setFullYear(x), setMonth(x), ...*** – Imposta anno (4 cifre), mese (0-11), ecc.
  - ***getDay()*** – Restituisce il giorno (0-6; 0 = domenica, 1 = lunedì, ecc.)
  - ***getTime()*** – Restituisce il numero di millisecondi dal 01-01-1970
  - ***setTime(x)*** – Imposta i millisecondi dal 01-01-1970
  - ***parse(x)*** – A partire da data in formato stringa, restituisce il numero di millisecondi dal 01-01-1970
  - ***toString()*** – Converte l'oggetto *Date* in stringa

JavaScript
Oggetto Math

- Oggetto predefinito *Math*: costanti e funzioni matematiche

- Propriertà
  - ***E, PI*** – Numero di Neplero e Pi greco
- Metodi
  - ***abs(x)*** – Restituisce il valore assoluto di *x*
  - ***floor(x), ceil(x), round(x)*** – Arrotonda *x* ad un intero
  - ***sin(x), cos(x), tan(x), ...*** – Restituisce il seno, coseno, tang... di *x*
  - ***exp(x), log(x)*** – Funzioni esponenziale and logaritmica
  - ***max(x,y), min(x,y)*** – Restituisce il valore più alto o basso tra *x* e *y*
  - ***pow(x,y)*** – Restituisce il valore di *x* elevato alla potenza di *y*
  - ***random()*** – Restituisce un numero casuale tra 0 e 1
  - ***sqrt(x)*** – Restituisce la radice quadrata di *x*

DOM

DOM
Document Object Model

Alberto Ferrari

DOM
Document Object Model

- API per documenti html e xml
  - Non è una particolare applicazione o prodotto
  - È una interfaccia che i browser devono implementare per conformarsi allo standard W3C DOM
- Per uno sviluppatore, significa due cose
  - Fornisce una rappresentazione strutturata del documento
  - Definisce come accedere a questa struttura da script, permettendo di gestire le pagine web come un gruppo strutturato di nodi
- Essenzialmente, connette le pagine web agli script e ai linguaggi di programmazione

Alberto Ferrari

DOM
Document Object Model

- Es. tutti i browser che implementano DOM devono restituire tutti gli elementi <*P*> in una pagina HTML come array di nodi quando viene invocato il metodo *getElementsByTagName* del documento:

```
paragraphs = document.getElementsByTagName("p");
// paragraphs[0] is the first <p> element
// paragraphs[1] is the second <p> element, etc.
alert(paragraphs[0].nodeName);
```

- **Esempio**

Alberto Ferrari

DOM
Oggetto Window

- Proprietà
  - ***document*** – Oggetto documento
  - ***event*** – Evento attuale
  - ***location*** – Url attuale
  - ***name*** – Nome della finestra
  - ***navigator*** – Oggetto navigator (caratteristiche del browser)
  - ***self, parent, top*** – Frame attuale, genitore o radice
  - ***status*** – Messaggio della barra di stato
- Metodi
  - ***alert(msg), confirm(msg), prompt(msg)*** – Visualizza una finestra di dialogo
  - ***open(url, name, ...), close()*** – Apre o chiude una finestra
  - ***setTimeout(expr, millis)*** – Valuta una espressione dopo un intervallo di tempo specificato
- Esempio

Alberto Ferrari

DOM
Oggetto Navigator

- Informazioni sul browser usato dall'utente
- Proprietà
  - ***appName*** – Nome del browser
  - ***appVersion*** – Piattaforma e versione del browser
  - ***browserLanguage*** – Lingua del browser
  - ***cookieEnabled*** – Cookie abilitati, o no?
  - ***cpuClass*** – Stringa che identifica la classe della CPU
  - ***onLine*** – Il sistema è on-line, o no?
  - ***platform*** – Piattaforma del browser
  - ***systemLanguage*** – Lingua di default del sistema
  - ***userAgent*** – Agente-utente HTTP
  - ***userLanguage*** – Attuale lingua impostata dall'utente
- Esempio

Alberto Ferrari

DOM
Oggetto Document

- Documento html contenuto nella finestra
- Proprietà
  - ***anchors, applets, forms, images, links*** – Collezione di tutti gli elementi anchor (...) del documento
  - ***cookie*** – Cookies del documento
  - ***body*** – Elemento body o frameset
  - ***title*** – Titolo del documento
- Metodi
  - ***open(), close()*** – Apre o chiude un documento
  - ***write(text)*** – Scrive testo su un documento
  - ***getElementById(id)*** – Trova un elemento dato il suo id
  - ***createElement(type)*** – Crea un elemento del tipo specificato
- Esempio , Esempio

Alberto Ferrari

DOM
Element

- Tutti gli elementi condividono una interfaccia comune
  - Ci sono interfacce più specializzate per oggetti particolari
  - L'elemento *body*, per esempio, ha funzioni e proprietà extra
- Proprietà
  - ***childNodes*** – Array dei nodi figlio dell'elemento
  - ***innerHTML*** – Tutto il contenuto, con il markup, all'interno di un dato elemento
  - ***style*** – Il blocco di regole di stile per l'elemento corrente
- Metodi
  - ***appendChild(element)*** – Aggiunge il nodo specificato nella lista di nodi del documento attuale
  - ***getElementsByTagName(name)*** – Restituisce la collezione di tutti gli elementi con uno specifico nome di tag
  - ***getAttribute(name), setAttribute(name, value)*** – Restituisce o modifica un attributo dell'elemento

DOM
Oggetto Form

- Proprietà
  - ***action*** – Attributo action del form
  - ***elements*** – collezione degli elementi del form
  - ***length*** – Numero di elementi del form
  - ***method*** – Metodo http per sottomettere il form
  - ***name*** – Nome del form
  - ***target*** – Frame o finestra dove visualizzare la risposta
- Metodi
  - ***reset()*** – Cancella i valori inseriti dall'utente nel form
  - ***submit()*** – Sottomette il form
- Esempio

Esempi
Validare un form

```
<html>
<head>
  <script language="JavaScript"><!--
    function validate() {
      field = document.getElementById('id1');
      if (field.value.length > 0) {
        return true;
      }
      else {
        alert('Text field empty!');
        return false;
      }
    }
  //--></script>
</head>
<body>
  <form name="form1" onsubmit="return validate()">
    <input id="id1" name="field1" type="text" />
    <input type="submit" />
  </form>
</body>
</html>
```

- Esempio

Esempi
Nascondere un elemento

```
<html>
<head>
  <script language="JavaScript"><!--
  function hide(elm) {
    document.getElementById(elm).style.visibility = 'hidden';
  }
  function show(elm) {
    document.getElementById(elm).style.visibility = 'visible';
  }
//--></script>
</head>
<body>
  <div id="id1" onclick="hide('id3')">Hide</div>
  <div id="id2" onclick="show('id3')">Show</div>
  <div id="id3">Some text</div>
</body>
</html>
```

- Esempio

Esempi
Muovere un elemento

```
<html><head>
  <script language="JavaScript"><!--
  var t = null; var x = 0;
  function move() {
    x += 5; if (x > 500) x = 0;
    document.getElementById('id3').style.left = x + 'px';
    t = window.setTimeout('move()', 500);
  }
  function home() {
    window.clearTimeout(t); x = 0; t = null;
    document.getElementById('id3').style.left = x + 'px';
  }
  //--></script>
</head>
<body>
  <div id="id1" onclick="if (t == null) move()">Move</div>
  <div id="id2" onclick="home()">Home</div>
  <div id="id3" style="position:absolute;left:0;">Some text</div>
</body></html>
```


Esempi
Creare un elemento

```
<html>
<head>
  <script language="JavaScript"><!--
  function create() {
    var id2 = document.getElementById('id2');
    id2.innerHTML = 'Enter your <b>password</b>: ';
    var newElement = document.createElement('input');
    newElement.setAttribute('type', 'password');
    id2.appendChild(newElement);
  }
  //--></script>
</head>
<body>
  <div id="id1" onclick="create()">Create</div>
  <form id="id2">Some text</form>
</body>
</html>
```