Sapienza Università di Roma
corso di laurea in Ingegneria dei sistemi informatici
corso di laurea in Ingegneria informatica e automatica

Linguaggi per il Web

a.a. 2014/2015

Parte 3
Javascript

Luigi Dragone, Riccardo Rosati

Introduzione

- **Javascript** è un linguaggio si scripting **lato client** utilizzato per rendere dinamico il codice HTML.
- Il documento HTML è generato staticamente.
- Il codice Javascript è immerso nel documento HTML ma viene **eseguito dinamicamente** solo al momento della richiesta del web client (browser).
- Principali usi:
  - posizionamento dinamico degli oggetti;
  - validazione campi dei moduli;
  - effetti grafici;
  - …

Introduzione

- JavaScript è un linguaggio di programmazione di tipo script:
  - Non è compilato
  - E’ ad alto livello
- La sua caratteristica principale è di essere nato espressamente per il WWW (è stato introdotto da Netscape nel 1995) e di essere supportato dalla maggior parte dei browser in uso.

Introduzione

- Non può essere adoperato per costruire programmi complessi o con particolari requisiti di prestazioni, tuttavia può essere impiegato per implementare velocemente piccole procedure e controlli.
- Il suo campo d'impiego naturale è la gestione sul web client (browser) di alcuni elementi delle applicazioni per WWW.

PRIMA PARTE

Il linguaggio JavaScript

Sintassi

- Un programma JavaScript è composto da una sequenza di istruzioni terminate dal punto e virgola (`;`)

- Un insieme di istruzioni delimitate da parentesi graffe (`{` e `}`) costituiscono un blocco.

Sintassi

- Un **identificatore** è una sequenza di simboli che identifica un elemento all'interno del programma
- Un identificatore può essere composto da lettere e numeri, ma:
  - non deve contenere elementi di punteggiatura o spazi, e
  - deve cominciare con una lettera.

Sintassi

- Questi sono identificatori validi
  - `nome`
  - `Codice_Fiscale`
  - `a0`

- Questi sono identificatori non validi
  - `0a`
  - `x.y`
  - `Partita IVA`

Sintassi

- Il linguaggio è **case-sensitive**, ovvero distingue le lettere maiuscole dalle minuscole
- I commenti sono deliminati da `/*` e `*/`
- Il simbolo `//` indica che il testo seguente sulla medesima riga è un commento

Variabili

- Una variabile è un'area di memoria contenente informazioni che possono "variare" nel corso dell'esecuzione del programma.
- Una variabile è caratterizzata da un nome identificativo e dal tipo dell'informazione contenuta.

Definizione di variabili

- Prima di essere adoperata una variabile deve essere definita

```
var eta;
```

- La definizione stabilisce il nome della variabile, mentre il tipo dipende dall'assegnazione

```
eta = 35;    //intero
nome = "Mario Rossi"; //stringa
```

Definizione di variabili

- Il tipo di una variabile dipende dall'ultima assegnazione, quindi una variabile può cambiare tipo nel corso del suo ciclo di vita

```
x = 10; //intero
…
x = "a"; //stringa
```

- JavaScript è un linguaggio "debolmente tipato"

Tipi dato predefiniti

- Number
- Boolean
- Null
- String
- Date
- Array

Tipo Number

- Una variabile di tipo Number assume valori numerici interi o decimali

```
var x = 10;     //valore intero
var y = -5.3;   //valore decimale
```

- Sono definite le operazioni aritmetiche fondamentali ed una serie di funzioni matematiche di base

Tipo Boolean

- Una variabile di tipo Boolean assume i soli valori della logica booleana vero e falso

```
var pagato = true;   //valore logico vero
var consegnato = false; //valore logico falso
```

- Sono definite le operazioni logiche fondamentali (AND, OR e NOT)

Tipo Boolean

AND:

X	Y	X AND Y
true	true	true
true	false	false
false	true	false
false	false	false

Tipo Boolean

OR:

X	Y	X OR Y
true	true	true
true	false	true
false	true	true
false	false	false

Tipo Boolean

NOT:

X	NOT X
true	false
false	true

Tipo Null

- Si tratta di un tipo che può assumere un unico valore

```
var a = null;
```

- Serve ad indicare che il contenuto della variabile è non significativo

```
var sesso = “f”;
var militesente = null;
```

Tipo String

- Una variabile di tipo String contiene una sequenza arbitraria di caratteri
- Un valore di tipo Stringa è delimitato da apici (‘ ’) o doppi-apici (“”)

```
var nome = “Mario Rossi”;
var empty = “”; //Stringa vuota
var empty2 = new String(); //Stringa vuota
var str = ‘Anche questa è una stringa’;
var str2 = new String(“Un’altra stringa”);
```

Tipo Date

- Una variabile di tipo Date rappresenta un istante temporale (data ed ora)

```
var adesso = new Date();
var natale2012 = new Date(2012,11,25);
var capodanno2013 = new Date("Gen 1
  2013");
```

- E' definito l'operatore di sottrazione (−) tra due date che restituisce la differenza con segno espressa in millisecondi

Tipo Array

- Un array è un vettore monodimensionale di elementi di tipo arbitrario

```
var v = new Array(); //Vettore vuoto
var w = new Array("Qui", "Quo", "Qua");
var u = new Array("Lun", "Mar", "Mer",
  "Gio", "Ven", "Sab", "Dom");
```

- Non è necessario specificare la dimensione

Assegnazione

- L'assegnazione è l'operazione fondamentale nei linguaggi di programmazione imperativa

  `id = expr`

- Dapprima viene "valutata" l'espressione **`expr`**, quindi il risultato viene "assegnato" alla variabile **`id`**

Assegnazione

- Si consideri il seguente frammento di codice

```
var x = 10;
var y = 7;
var z = 3*(x-y);
```

- Al termine dell'esecuzione la variabile `z` assume il valore 9

Espressioni

- Un’espressione è composta da
  - identificatori di variabili

    `x, nome, eta, importo, …`
  - costanti

    `10, 3.14, “<HTML>”, …`
  - operatori
  - parentesi (`(` e `)`) per alterare le regole di precedenza tra operatori

Operatori aritmetici

- Operano tra valori (costanti o variabili) numerici
  - Somma (`1+1`)
  - Sottrazione (`3-5`)
  - Moltiplicazione (`3*4`)
  - Divisione (`6/4`)
  - Modulo (`6%5`)
  - Cambiamento di segno (`-3`)

Operatori di pre/post incremento/decremento

- Derivano dal linguaggio C (e sono presenti in C++ e Java)
- Alterano e restituiscono il valore di una variabile
- Consentono di scrivere codice compatto ed efficiente

Operatori di pre/post incremento/decremento

```
var x = 10;
var y = x++; //y=10 e x=11
var z = ++x; //z=12 e x=12
var w = x--; //w=12 e x=11
var v = --x; //v=10 e x=10
```

Operatori su stringhe

- L'unica operazione possibile in un'espressione è la concatenazione di stringhe

```
nomeCompleto = titolo + " " + nome + " "
  + cognome
indirizzo = recapito + ", " + numCivico +
  " " + CAP + " – " + citta + " (" + prov
  + ")"
```

Operatori su stringhe

- Esiste una forma più compatta per esprimere l'accodamento
- L'istruzione seguente

```
str = str + newStr;
```

è equivalente a

```
str += newStr;
```

Operatori su vettori

- L'operatore definito sui vettori è l'accesso ad un determinato elemento dato l'indice

```
v[3] = 10;
a[i] = b[i] * c[i];
p = v[1];
```

- Il primo elemento di un vettore ha sempre l'indice 0 (come in C/C++ e Java)

Operatori sui vettori

- Se si assegna un valore ad un elemento non presente questo viene creato

```
var v = new Array();
v[0] = 1;
```

- Se si accede al valore di un elemento non presente si ottiene un valore indefinito

```
var v = new Array();
var a = v[0];
```

Operatori relazionali

- Confrontano due valori e restituiscono l'esito come valore booleano
  - Uguaglianza (==)
  - Disuguaglianza (!=)
  - Minore (<)
  - Maggiore (>)
  - Minore o uguale (<=)
  - Maggiore o uguale (>=)

Operatori logici

- Operano tra valori (costanti o variabili) booleani
  - AND (&&)
  - OR (||)
  - NOT (!)
- Solitamente gli operandi sono l'esito di un confronto

Condizioni

- L’utilizzo di operatori relazionali e logici consente di formulare delle condizioni che possono essere utilizzate per controllare l’esecuzione del programma

```
(metodoPagamento==“contrassegno”)&&
  (!residenteInItalia)
```

Controllo dell’esecuzione

- L’esecuzione di un programma è generalmente sequenziale
- Tuttavia in determinate “condizioni” può essere necessario eseguire solo alcune istruzioni, ma non altre, oppure ripetere più volte un’operazione

Istruzione if

- L'istruzione **`instr`** viene eseguita solo se la condizione **`cond`** risulta vera

```
if(cond)
  instr
```

- L'istruzione **`instr`** può essere sostituita da un gruppo di istruzioni tra parentesi graffe (`{` e `}`)

Istruzione if

- Una costruzione alternativa prevede la presenza di una seconda istruzione da eseguire nel caso la condizione risulti falsa

```
if(cond)
  instr_then
else
  instr_else
```

Istruzione if

```
if(scelta==“NO”) {
   // Se la scelta è NO
   …
} else {
   // Altrimenti
   …
}
```

Istruzione for

- Viene dapprima eseguita l'istruzione **`init`**, quindi l'istruzione **`instr`** viene ripetuta finché la condizione **`cond`** risulta vera, dopo ogni ripetizione viene eseguita l'istruzione **`next`**

```
for(init; cond; next)
  instr
```

Istruzione for

- Inizializzare a `0` gli `n` elementi del vettore `a`

```
for(var i=0; i<n; i++)
   a[i]=0;
```

- Copiare gli `n` elementi del vettore `a` nel vettore `b`

```
for(var i=0; i<n; i++)
   b[i]=a[i];
```

Istruzione while

- L'istruzione **`instr`** viene eseguita finché la condizione **`cond`** risulta essere verificata

```
while(cond)
  instr
```

- Un'istruzione for può essere espressa come

```
init;
while(cond) {instr; next;}
```

Funzioni

- Una funzione è un elemento di un programma che calcola un valore che dipende "funzionalmente" da altri

  $y \leftarrow f(x)$

  $y \leftarrow log_{10}(x)$
- L'utilizzo delle funzioni nella programmazione strutturata aumenta la modularità e favorisce il riutilizzo

Definizione di funzioni

- In JavaScript è possibile definire una o più funzioni all'interno di un programma

```
function name(arg0, arg1, …, argn-1) {
   …
}
```

- La funzione definita è identificata da **`name`** e dipende dagli argomenti **arg_0, arg_1, …, arg_{n-1}**

Definizione di funzioni

- Somma di due numeri

```
function somma(a, b) {
   return a+b;
}
```

- La funzione viene “invocata” all’interno di un’espressione

```
var s = somma(1, 2);
```

Invocazione di funzioni

- Quando una funzione viene invocata gli argomenti sono inizializzati con i valori specificati
- Quindi si procede con l'esecuzione delle istruzioni costituenti la funzione
- L'istruzione `return` restituisce il valore calcolato al chiamante

Invocazione di funzioni

- La chiamata

  ```
  x = somma(1, 2)
  ```

  inizializza gli argomenti `a` e `b` rispettivamente ai valori 1 e 2
- L'istruzione

  ```
  return a+b;
  ```

  valuta l'espressione e restituisce il risultato (3) che viene assegnato alla variabile `x`

Variabili locali e globali

- All’interno di una funzione è possibile definire delle variabili “confinate” all’interno della funzione stessa
- Tali variabili, dette **locali**, sono create all’atto dell’invocazione della funzione e sono distrutte al termine dell’esecuzione
- Il loro valore non è accessibile dall’esterno della funzione
- Ogni argomento di una funzione è una variabile locale definita implicitamente

Variabili locali e globali

- Le variabili definite all'esterno di una funzione sono denominate, invece, **globali**
- A differenza delle variabili locali, le variabili globali sono accessibili da qualsiasi punto del programma, anche dall'interno di una funzione, sempre che in quest'ultima non sia stata definita una variabile locale con lo stesso nome

Variabili locali e globali

- Si consideri il seguente frammento di codice

```
function f(…) {
        …
        var x = 1;
        …
}
var x = –1;
…f(…);
```

- La variabile globale `x` continua a valere -1

Procedure

- Una funzione che non restituisce valori viene detta **procedura** (in analogia con il Pascal)
- Una procedura agisce in genere su variabili globali (*side-effect*)

Funzioni predefinite

- In JavaScript sono presenti alcune **funzioni predefinite**
  - `isNaN(v)` verifica se `v` non è un numero
  - `isFinite(v)` verifica se `v` è finito
  - `parseFloat(str)` converte `str` in un numero decimale
  - `parseInt(str)` converte `str` in un numero intero

Oggetti

- Un oggetto è un elemento caratterizzato da uno stato rappresentato mediante proprietà e da un insieme di azioni (o metodi) che può eseguire
- Oggetti caratterizzati dagli stessi metodi e dalle stesse proprietà, ma non dallo stesso stato, sono detti della stessa classe

Oggetti

- JavaScript è un linguaggio orientato agli oggetti, tuttavia non possiede il costrutto di classe
- Molti tipi di dato fondamentali sono, in effetti, degli oggetti (String, Date, Array,…)

Proprietà e metodi

- Una proprietà di un oggetto è assimilabile ad una variabile

```
cliente.nome = “Carlo Bianchi”;
x = ordine.aliquotaIVA;
```

- Un metodo, invece, è simile ad una funzione

```
tot = ordine.calcolaTotale();
```

Proprietà e metodi

- Esistono due sintassi alternative per accedere alle proprietà degli oggetti

```
oggetto.proprieta
oggetto[“proprieta”]
```

- La seconda è utile quando il nome della proprietà viene determinato durante l’esecuzione del programma

Oggetti di tipo String

- Proprietà
  - `length` lunghezza della stringa
- Metodi
  - `charAt(pos)` carattere alla posizione `pos`
  - `subString(start, end)` sottostringa dalla posizione `start` alla posizione `end`
  - `toUpperCase()`/`toLowerCase()` converte la stringa in maiuscolo/minuscolo
  - `indexOf(str, pos)` posizione della prima occorrenza della string `str` cercata a partire dalla posizione `pos`

Oggetti di tipo Array

- Proprietà
  - `length` lunghezza del vettore
- Metodi
  - `sort()` ordina gli elementi del vettore
  - `reverse()` inverte l'ordine degli elementi del vettore

Oggetti di tipo Date

- Metodi
  - `get`*`XXX`*`()` restituisce il valore della caratteristica *`XXX`* della data (es. *getFullYear()*).
  - `set`*`XXX`*`(val)` imposta il valore della caratteristica *`XXX`* della data (es. *setFullYear(2013,1,1)*);
  - `toString()` restituisce la data come stringa formattata

Oggetti di tipo Date

Nome caratteristica	**Significato**
Date	Giorno del mese
Day	Giorno della settimana
FullYear	Anno
Minutes	Minuti
Hours	Ore
Month	Mese
Seconds	Secondi
Time	Tempo (hh:mm:ss)

Oggetto Math

- Proprietà
  - `E` costante di Eulero
  - `PI` pi greco
- Metodi
  - `abs(val)` valore assoluto
  - `ceil(val)`/`floor(val)` troncamento
  - `exp(val)` esponenziale
  - `log(val)` logaritmo
  - `pow(base, exp)` elevamento a potenza
  - `sqrt(val)` radice quadrata

SECONDA PARTE

Uso di JavaScript per il World Wide Web

Integrazione con i browser web

- La caratteristica principale di JavaScript è di essere “integrabile” all’interno delle pagine web
- In particolare consente di aggiungere una logica procedurale alle pagine rendendole “dinamiche”
- A differenza di altre tecnologie, JavaScript funziona completamente sul client

Integrazione con i browser web

- I campi di impiego principali sono
  - Validazione dell'input dell'utente e controllo dell'interazione
  - Effetti visivi di presentazione
- JavaScript è supportato da tutti i browser più diffusi

Integrazione con i browser web

- Il linguaggio JavaScript è standard (ECMA-262), tuttavia
  - Solo le versioni più recenti dei browser sono conformi integralmente allo standard
  - Mentre le versioni più vecchie utilizzano versioni proprietarie del linguaggio (JScript)
  - Inoltre, l'integrazione tra browser e linguaggio non è standardizzata

Dynamic HTML (DHTML)

- L'integrazione degli script all'interno di una pagina web avviene in due modi
  - Associando funzioni JavaScript agli eventi che si intende gestire
  - Accedendo dalle funzioni JavaScript alle proprietà degli oggetti che costituiscono la pagina

Dynamic HTML (DHTML)

- Una pagina “dinamica” (DHTML) è una pagina HTML contenente codice JavaScript associato a determinati eventi che implementa specifiche funzionalità
- Non bisogna confondere una pagina DHTML con una pagina generata dinamicamente dal server

Modello ad oggetti

- Un browser web esporta verso JavaScript un modello ad oggetti della pagina e dell'"ambiente" in cui la pagina è visualizzata
- Una funzione JavaScript adopera tali oggetti invocando i metodi e accedendo alle proprietà
- Il modello ad oggetti, a differenza del linguaggio, non è standard

Oggetto navigator

- L'oggetto `navigator` rappresenta l'istanza del browser in cui lo script è in esecuzione
- Proprietà
  - `appCodeName` codice identificativo del browser
  - `appName` nome del browser
  - `appVersion` numero di versione

Oggetto window

- Questo oggetto rappresenta la finestra in cui il documento corrente viene visualizzato
- Una funzione può accedere alle proprietà della finestra corrente, ma può creare e manipolare nuove finestre (pop-up)

Oggetto window

- Proprietà
  - `title` titolo della finestra
  - `statusbar` testo mostrato sulla barra di stato
  - `location` URL del documento visualizzato
  - `outerHeight`, `outerWidth` dimensioni esterne
  - `innerHeight`, `innerWidth` dimensioni interne

Oggetto window

- Modificare il titolo della finestra corrente

```
window.title = “Questo è il nuovo
  titolo”;
```

- Accedere ad un nuovo documento

```
window.location =
  “http://www.yahoo.com/”;
```

- Calcolare l’area in pixel della finestra

```
var area = window.innerWidth *
  window.innerHeight;
```

Oggetto window

- Metodi
  - `open(location, title)` apre una nuova finesta
  - `alert(message)` visualizza il messaggio in una finestra di dialogo (utile per il debug)
  - `confirm(message)` visualizza il messaggio e richiede una conferma all'utente
  - `moveTo(x, y)` sposta la finestra alle coordinate indicate
  - `resizeTo(width, height)` dimensiona la finestra

Oggetto window

- Creazione e posizionamento di una nuova finestra

```
var w = window.open(“http://www.
  google.com/”, “Google”);
w.moveTo(0, 0);
```

Oggetto window

- Visualizzazione di un messaggio

```
window.alert(“Attenzione si è
  verificato un errore”);
```

- Visualizzazione del browser in uso

```
window.alert(“Sei connesso con ” +
  navigator.appName + “ versione ” +
  navigator.version);
```

Oggetto window

- Richiesta conferma all'utente

```
if(confirm("Vuoi proseguire con
  l'operazione?")) {
   //L'utente ha risposto SI
   …
  } else {
   //L'utente ha risposto NO
   …
  }
```

Oggetto history

- Rappresenta la sequenza di pagine visitate dall'utente
- Tale sequenza è rappresentata mediante un vettore
- Metodi
  - `back()` torna alla pagina precedente
  - `forward()` passa alla pagina successiva

Oggetto document

- Rappresenta il documento HTML che costituisce la pagina visualizzata
- Non è possibile accedere a tutti gli elementi del documento
- Tuttavia è possibile accedere agli elementi dei moduli (form) ed alle proprietà di visualizzazione

Oggetto document

- Inoltre, è possibile costruire on-the-fly il documento prima che questo sia stato completamente caricato e visualizzato

Oggetto document

- Proprietà
  - `bgColor` colore dello sfondo
  - `fgColor` colore del testo
  - `forms` vettore dei moduli presenti nella pagina
  - `title` titolo del documento
  - `URL` indirizzo del documento
- Metodi
  - `write(string)` accoda `string` al documento, serve per la costruzione on-the-fly

Oggetto document

- Supponendo che nel documento HTML sia definito un modulo di nome *modulo*

```
<FORM NAME="modulo"…>

…

</FORM>
```

Oggetto document

- Si può accedere a tale oggetto in due diversi modi

  `document.forms[“`*`modulo`*`”];`

  `document.`*`modulo`*`;`

- Ciò è possibile, in generale, per tutti gli elementi del documento con un attributo `NAME`

Oggetto document

- Dal momento che la proprietà `forms` è di tipo `Array` è possibile accedervi anche tramite l'indice numerico dell'elemento

```
for(var i=0; i<document.forms.length; i++)
  {
     //Accedi a document.forms[i]
     ... = document.forms[i];
     ...
   }
```

Oggetto Form

- Un oggetto di questo tipo corrisponde ad un modulo all'interno di una pagina HTML
- Tramite le proprietà di questo oggetto è possibile accedere ai diversi elementi (o controlli) del modulo (inputbox, listbox, checkbox, ecc.)

Oggetto Form

- Proprietà
  - `action` valore dell'attributo `ACTION`
  - `elements` vettore contenente gli elementi del modulo
  - `length` numero di elementi del modulo
  - `method` valore dell'attributo `METHOD`
  - `target` valore dell'attributo `TARGET`

Oggetto Form

- Metodi
  - `reset()` azzera il modulo reimpostando i valori di default per i vari elementi
  - `submit()` invia il modulo

Oggetto Form

- Supponendo che l'i-esimo elemento di un modulo `mod` sia denominato `nome_i` è possibile farvi riferimento in 3 modi diversi

```
document.mod.elements[i-1];
document.mod.elements["nome_i"];
document.mod.name_i;
```

- Attenzione l'indice del primo elemento di un vettore è sempre 0 (quindi l'i-esimo elemento ha indice i-1)

Elementi dei moduli

- All’interno di un modulo possono comparire diversi tipi di elementi, corrispondenti ai vari costrutti HTML
- Ogni tipo ha proprietà e metodi specifici, per una trattazione approfondita si rimanda alla guida di riferimento del modello ad oggetti implementato dal browser

Elementi dei moduli

HTML	JavaScript
<INPUT TYPE="text">	Text
<TEXTAREA></TEXTAREA>	Textarea
<SELECT></SELECT>	Select
<INPUT TYPE="checkbox">	Checkbox
<INPUT TYPE="radio">	Radio
<INPUT TYPE="button">	Button

Elementi dei moduli

- Tutti i tipi di elementi possiedono le seguenti proprietà
  - `name` nome dell'elemento
  - `value` valore corrente dell'elemento
- Gli elementi di tipo `Input` possiedono la proprietà `defaultValue` che contiene il valore predefinito del campo (attributo `VALUE` del tag HTML)

Elementi dei moduli

- Gli elementi di tipo `Radio` e `Checkbox` possiedono la proprietà `checked` che indica se l'elemento è stato selezionato
- Gli elementi di tipo `Select` possiedono la proprietà `selectedIndex`, che contiene l'indice dell'elemento selezionato nella lista, e la proprietà `options`, che contiene il vettore delle scelte dell'elenco

Elementi dei moduli

- E’ possibile modificare i valori contenuti negli elementi dei moduli
- Pertanto è possibile utilizzare questi elementi anche per fornire risultati all’utente
- Se un elemento ha scopi esclusivamente di rappresentazione può essere marcato come `READONLY`

Esempio: modulo di iscrizione

- Il modulo deve raccogliere i dati su un utente che vuole sottoscrive un certo servizio
- Per ogni utente deve richiedere
  - nominativo, età, sesso
  - se desidera ricevere informazioni commerciali
  - il servizio cui desidera iscriversi

Esempio: modulo di iscrizione

- Il modulo deve suggerire un'età di 18 anni ed il consenso all'invio di informazioni commerciali
- I servizi disponibili sono denominati "Servizio 1", "Servizio 2" e "Servizio 3"
- Il modulo deve suggerire la sottoscrizione al primo servizio

Esempio: modulo di iscrizione

- Per ogni dato si determina il tipo di elemento da inserire nel modulo
  - nominativo ed età con elementi di tipo Text
  - sesso con un elementi di tipo Radio
  - infoComm con un elemento di tipo Checkbox
  - servizio con un elemento di tipo Select
- Si deve, inoltre, inserire il pulsante di invio del modulo

Esempio: modulo di iscrizione

- Per gestire la presentazione si adopera una tabella HTML che presenta sulla prima colonna il nome del campo e nella seconda gli elementi corrispondenti

Esempio: modulo di iscrizione

```
<HTML>
<BODY>
<FORM NAME="iscrizione" ACTION="" METHOD="POST">
<TABLE>
…
```

Esempio: modulo di iscrizione

```
…
<!-- Nominativo -->
<TR>
   <TD>Nominativo</TD>
   <TD>
    <INPUT NAME="nominativo" TYPE="text" SIZE="40">
   </TD>
</TR>
…
```

Esempio: modulo di iscrizione

```
…
<!-- Eta' -->
<TR>
   <TD>Et&agrave;</TD>
   <TD>
    <INPUT NAME="eta" TYPE="text" SIZE="3" VALUE="18">
   </TD>
</TR>
…
```

Esempio: modulo di iscrizione

```
<!-- Sesso -->
<TR>
   <TD>Sesso</TD>
   <TD>
   <INPUT NAME="sesso" TYPE="radio" VALUE="M">M
   <INPUT NAME="sesso" TYPE="radio" VALUE="F">F
</TD>
</TR>
```

Esempio: modulo di iscrizione

```
…
<!-- Inform. commerciali -->
<TR>
   <TD>Inform. commerciali</TD>
   <TD>
     <INPUT NAME="infoComm" TYPE="checkbox" CHECKED>
   </TD>
</TR>
…
```

Esempio: modulo di iscrizione

```
<!-- Servizio -->
<TR>
   <TD>Servizio</TD>
   <TD>
   <SELECT NAME="servizio">
   <OPTION VALUE="s1" SELECTED>Servizio 1</OPTION>
   <OPTION VALUE="s2">Servizio 2</OPTION>
   <OPTION VALUE="s3">Servizio 3</OPTION>
   </SELECT>
   </TD>
</TR>
```

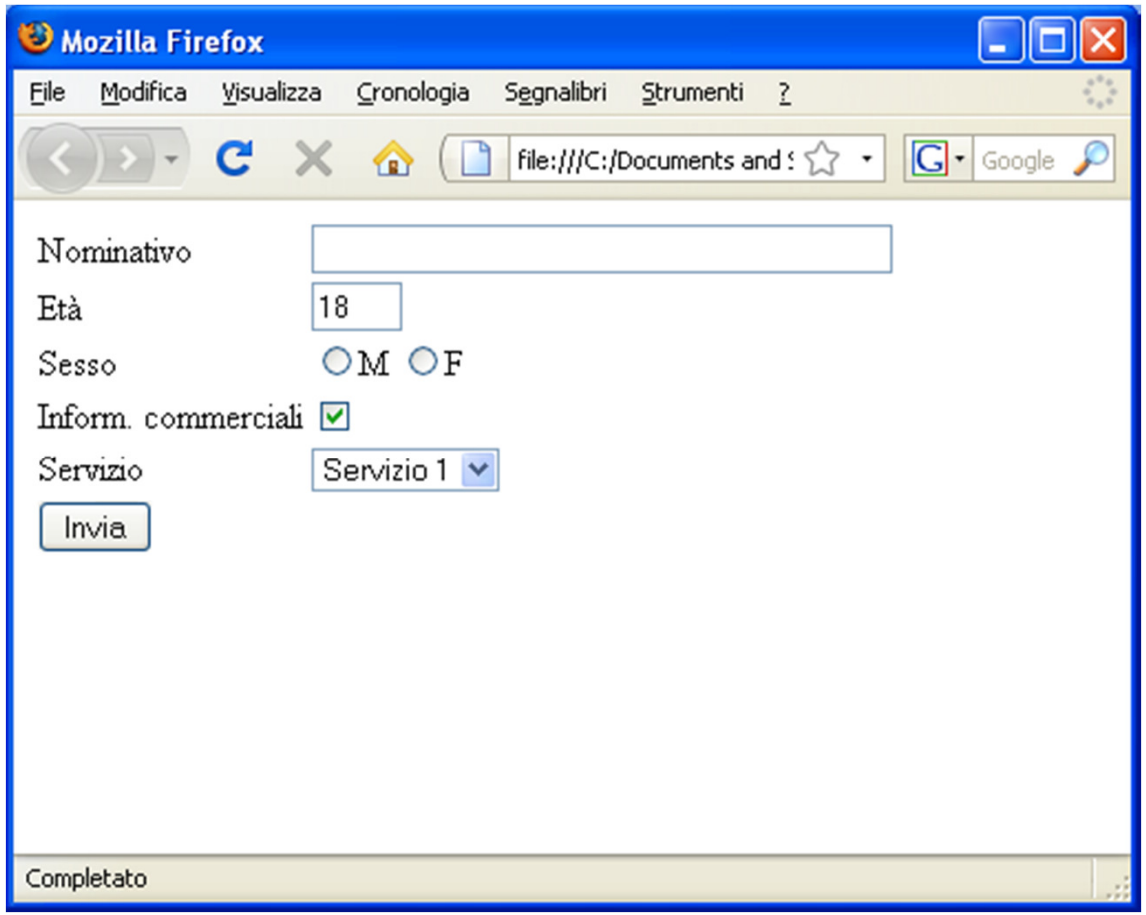
Esempio: modulo di iscrizione

```
…
<!-- Invio del modulo -->
<TR>
   <TD>
     <INPUT TYPE=“submit” VALUE=“Invia”>
   </TD>
</TR>
…
```

Esempio: modulo di iscrizione

```
…
</TABLE>
</FORM>
</BODY>
</HTML>
```

Esempio: modulo di iscrizione



Esempio: modulo di iscrizione

- Per accedere al nominativo immesso

  `document.iscrizione.nominativo.value`
- Per accedere all'età

  `document.iscrizione.eta.value`
- Per accedere al valore numerico dell'età

  `parseInt(document.iscrizione.eta.value)`

Esempio: modulo di iscrizione

- Per visualizzare un messaggio relativo alla scelta di riceve informazioni commerciali:

```
if(document.iscrizione.infoComm.checked)
   alert("Vuoi ricevere informazioni
    commerciali");
else
   alert("Non vuoi ricevere informazioni
    commerciali");
```

Eventi

- Ogni oggetto di un documento HTML “genera” degli **eventi** in risposta alle azioni dell’utente
- Ad esempio, l’evento `click` corrisponde al click del puntatore sull’oggetto
- Per gestire l’interazione con l’utente si associano funzioni JavaScript a particolari eventi

Eventi

- Gli eventi generati da un oggetto dipendono dal tipo di quest'ultimo
- Oggetti `Form`
  - `onSubmit` invio del modulo
  - `onReset` azzeramento del modulo
- Oggetti `Button`
  - `onClick` click del puntatore

Eventi

- Oggetti `Select`, `Text` e `Textarea`
  - `onChange` modifica del contenuto
  - `onFocus` selezione dell'elemento
- Oggetti `Radio` e `Checkbox`
  - `onClick` click del puntatore
  - `onFocus` selezione dell'elemento

JavaScript in documenti HTML

- Il codice JavaScript viene inserito all'interno di una pagina HTML delimitato dal tag `<SCRIPT>…</SCRIPT>`
- Per motivi di compatibilità con i vecchi browser il codice è incluso in un commento HTML `<!--…-->`

JavaScript in documenti HTML

- Generalmente il codice è incluso all'interno del'intestazione (<HEAD>) del documento
- Il codice viene eseguito **prima** della visualizzazione del documento.
- In questa sezione si procede con la definizione delle funzioni e delle variabili globali

JavaScript in documenti HTML

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript"
  LANGUAGE="javascript">
<!--
    function f(…);
    var v=…;
    …
//-->
</SCRIPT>
…
</HEAD>
…
</HTML>
```

JavaScript in documenti HTML

- Eventualmente il codice JavaScript può risiedere in un documento esterno

```
<SCRIPT TYPE="text/javascript"
   LANGUAGE="javascript"
   SRC="URL.js">
</SCRIPT>
```

- Ciò è utile per aumentare la modularità e "nascondere" all'utente il codice

JavaScript in documenti HTML

- Adoperando il metodo `write()` dell'oggetto document è possibile costruire in fase di caricamento parte del corpo del documento HTML (`<BODY>…</BODY>`)
- Tuttavia questo sistema è poco pratico e se ne sconsiglia l'impiego

Intercettazione eventi

- Per intercettare l'evento *E* di un tag *T* ed associare l'esecuzione di una funzione *f()*

  `<`*T* `on`*E*`=“return` *f()*`;”>`

- Se il risultato della valutazione della funzione è `false` viene interrotta l'esecuzione del commando corrente, ad esempio l'invio di un modulo

Intercettazione eventi

- Ad esempio, la funzione `valida()` verifica se i dati immessi nel modulo `modulo` sono corretti ed eventualmente procede con l'invio di quest'ultimo

```
<FORM NAME="modulo"
  onSubmit="return valida();" …>
…
</FORM>
```

Validazione modulo di iscrizione

- Nel modulo di sottoscrizione del servizio è necessario indicare il nominativo del sottoscrittore
- Quindi il valore del campo corrispondente non deve essere una stringa vuota

Validazione modulo di iscrizione

- Il modulo viene ridefinito come

```
<FORM NAME="iscrizione" ACTION=""
  METHOD="POST" onSubmit="return
  validaIscrizione();">
```

- Nell'intestazione del documento viene definita una funzione `validaIscrizione()`

Validazione modulo di iscrizione

```
function validaIscrizione() {
   if(document.iscrizione.nominativo.value==“”)
     {
      alert(“Nominativo obbligatorio.
        Impossibile procedere.”);
      return false;
      }
   … //Altri controlli
   return true;
}
```

Proprietà di visualizzazione

- Tra le proprietà degli elementi del modello ad oggetti del documento sono presenti alcune specifiche della modalità di presentazione all'utente degli elementi medesimi
- La possibilità di poter accedere e manipolare tali caratteristiche permette di utilizzare JavaScript per ottenere sofisticati effetti di presentazione visiva
- Purtroppo, queste proprietà sono specifiche dei singoli browser

Proprietà di visualizzazione

- Le proprietà di visualizzazione sono connesse con l'uso dei CSS
- Infatti, alcune proprietà degli stili possono essere modificate dinamicamente da funzioni JavaScript
- Tra le proprietà più utili per la creazione di effetti visivi abbiamo la visualizzazione ed il posizionamento degli elementi

Tecnologie collegate a JavaScript

- **JQuery**: libreria di funzioni JavaScript (standardizza gli oggetti messi a disposizione dall'interprete JavaScript)
- **JSON** (JavaScript Object Notation): formato per lo scambio dei dati tra Web client e Web server (alternativo a XML)
- **AJAX** (Asynchronous JavaScript and XML): tecnica di sviluppo di applicazioni Web

Tecnologie collegate a JavaScript

AJAX (Asynchronous JavaScript and XML): tecnica di sviluppo di applicazioni Web basata su:

- Utilizzo di JavaScript
- Uso di XML (o JSON) come formato per lo scambio dei dati tra web server e web client
- scambio di dati in background tra web client e web server (senza esplicito ricaricamento della pagina web)

Riferimenti

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
- http://msdn.microsoft.com/
- http://www.w3c.org/
- http://www.html.it/javascript/
- http://www.ecmascript.org
- http://jquery.com/