JavaScript

- Linguaggio definito da Netscape
- JScript: la versione MicroSoft (basata su ECMAScript)
- Serve ad arricchire una pagina HTML con codice da eseguirsi sul cliente

Un esempio

- *http://www.di.unipi.it/~ghelli/didattica/bdl/esercizi/ese4/menuSubmit.html :*
  - *Risorse del corso->Materiale esercizi->ese4->menuSubmit.html*
- Provate a selezionare un esame
- La form si auto-sottomette per aggiornare i dati degli studenti

Un altro esempio

- *http://www.di.unipi.it/~ghelli/didattica/bdl/esercizi/ese4/menulocal.html :*
  - *Risorse del corso->Materiale esercizi->ese4->menulocal.html*
- Provate a selezionare un esame
- La form si auto-aggiorna, senza effettuare nessuna submit dietro le quinte, perché contiene le informazioni necessarie

Un altro esempio

- *http://www.di.unipi.it/~ghelli/didattica/bdl/esercizi/ese4/javascript.html :*
  - *Risorse del corso->Materiale esercizi->ese4->javascript.html*
- Provate a:
  - Selezionare un'opzione del combo box
  - Inserire un valore nel campo accanto
  - Scrivere una stringa nel campo auto-incrementante

Variabili, Operatori, Commenti

- Variabili con tipo, ma non dichiarato (e conversione implicita a string)
  - var x = 5
  - var s = "luigi"
  - s + x -> "luigi5"
- Tipi: numerici, stringhe, bool, funzioni, oggetti, null
- Identificatori: lettere+_, case sensitive, anche interi
- Terminazione dei comandi: newline, ;, o entrambi.
- Operatori del C: +,-,*,/,%, &&,||, ==, !=, <, >
- && ed || sono valutati in modo ordinato.
- Commenti: da // a fine linea (consigliato) e tra /* e */,

Costanti

```
• 3.9                  // numeric literal
• "Hello!"             // string literal
• "Perche'"            // string literal
• 'Value = "aa"'       // string literal
• false                // boolean literal
• null                 // literal null value
• {x:1, y:2}           // Object literal
• [1,2,3]              // Array literal
• function(x){return x*x;}
                       // function literal
```

Coercion

- Stringhe, booleani, e interi sono convertiti mutuamente se necessario
- Ad esempio:
  - “a” + 1 -> “a1”
- parseInt(“123”) si usa per convertire una stringa in un intero (accetta anche: 123abc)

Esempio di codice

```
var parsedF = parseInt(document.Forma1.Anno.value);
if (isNan(parsedF)) {
  alert(document.Forma1.Anno.value +
                        "is not an integer!");
} else {
  document.Forma1.Anno.value = parsedF;
}
```

Comandi: if e while

- If:

```
if ( cond ) {
comandi
}
```

- Oppure:

```
if ( cond ) {
comandi
} else {
comandi
}
```

- While:

```
while ( cond ) {
comandi
}
```

For e Funzioni

- for:

```
for (init; cond; incr) {
comandi
}
```

- Funzioni:

```
function NOME (listaParams) {
body
}
```

- Parametri separati da virgole, valore ritornato con `return(valore);` i parametri sono passati per valore

Oggetti (array associativi)

- var studenti = {
  BDL : ["Marco", "Mario", "Maria"],
  ALG : ["Lucia", "Linda", "Luca"]
  };
- studenti["BDL"] => ["Marco", "Mario", "Maria"]
  - studenti.BDL abbrevia studenti[“BDL"]
- studenti ["BDL"][0] => "Marco"
- x = "BDL"; studenti[x][0] => "Marco"
- for (x in studenti) { x … studenti[x] … }

Stringhe

- Concatenazione: +
- Alcuni metodi:
  - stringa.length
  - stringa.substring(start,end)
  - stringa.substr(start,length)
  - stringa.charAt(index)
- JScript supporta le espressioni regolari

Eventi Gestiti dal Browser

- Di pagina:
  - loading, unloading
- Associati ai bottoni:
  - click, submit
- Associati ai campi di tipo text e select:
  - change
  - select: selezionare una porzione di testo (non nei componenti select)
  - focus/blur: rendere il campo pronto ad accettare input

Associare codice ed eventi

- Attributo *onEvent*, per i componenti di una form:
  - ```
    <INPUT TYPE="button" NAME="mycheck"
     VALUE="HA!"
     onClick="alert('I told you not to click me');">
    ```
  - Il valore dell'attributo è un pezzo di codice che gestisce l'evento
- Attributo *onSubmit*, per l'intera form; se la funzione ritorna false, la sottomissione non avviene:
  - ```
    <FORM NAME="formname" ...
     onSubmit="submithandler()">
    ```
- Per il documento, *onLoad*, *onUnload*:
  - ```
    <BODY onLoad="loadfunc()"
     onUnload="unloadfunc()">
    ```
    ```

Il Tag SCRIPT

- Meglio metterlo nello head
- Carica da file:

```
<SCRIPT LANGUAGE="JavaScript"
    SRC="jscode/click.js">
</SCRIPT>
```

- Codice immediato: tra <SCRIPT> e </SCRIPT>, meglio se commentato con <!-- -->:

```
<!--
function dontclickme()      {
     alert("I told you not to click me");
     return( false );
}
<!-- end script -->
```

Leggere e scrivere i campi di una form

- Se la form si chiama myForm, con un campo text chiamato myText, posso scrivere, in una funzione:
  - document.myForm.myText.value += 1;
- Oppure, nel tag del campo:
  - onChange = “this.value += 1”
- Per un campo select, posso accedere alla prima opzione scrivendo:
  - document.myForm.mySelect.options[0].value += 1;
- Posso accedere all’opzione corrente scrivendo:
  - document.myForm.mySelect.options[document.myForm.mySelect.selectedIndex].value += 1;

Accedere ai campi per Id

- Aggiungere un attributo ID all'elemento:
  - `<INPUT TYPE="button"`
    `ID="mycheck"`
    `NAME="mycheck"  ....`
- A questo punto, posso scrivere:
  - `var bottone = getElementById('mycheck')`
- Più semplice di:
  - `var bottone = document.myform.mycheck`

Alcune funzioni importanti

- Funzioni:
  - form.submit()
  - alert()
  - navigate() / location.href = url
- Attributi (element.attribute = …):
  - Name
  - Value
  - InnerHTML
  - Style
  - …

Una funzione che fa un controllo

```
function checkit() {
    var strval = document.myform.mytext.value;
    var intval = parseInt(strval);
    if ( 0 < intval && intval < 10 ) {
        return( true );
    } else {
        alert("Value " + strval + " out of range");
        return( false );
    }
}
```

Esecuzione dell'esercizio

- Copio il file .../*javascript.html* sotto ~/*public_html/*
- Rendo il file leggibile da tutti:
  - chmod o+rx ~/public_html/javascript.html
- Esploro la forma creata:
  - http://www.cli.di.unipi.it/~mioNomeAccount/javascript.html
- La modifico lavorando sul file:
  - ~/public_html/javascript.html

Creazione procedura con loadpsp.exe

- Eseguire le procedure createEsami.sql e insertEsami.sql
- Modificare il file load.bat con i propri dati
- Eseguire il file load.bat (modificato)
- Verificare con sqlDeveloper, che sia stata creata la procedura “MenuSubmit” nella sottodirectory ‘Procedures’

Creazione della procedura con SQL

- Caricare ed eseguire in sqldeveloper il file menuSubmitPl.sql
- Verificare con sqlDeveloper, che sia stata creata la procedura “MenuSubmitPl” nella sottodirectory ‘Procedures’
- Modificare il file provaMenu.html con il proprio nome account Oracle
- Testare le due procedure create tramite provaMenu.html

Esercizio

- Uso charAt per fare si che venga aggiunto % solo quando la stringa non finisce con %
- Uso un while su AnnoPrimaIscrizione.options[i].value per settare selected solo quando ….[i].value = anno
- Aggiungo una checkbox Laureato, per cui, solo quando viene selezionata, appare un campo text ‘DataLaurea’

Esercizio

- Aggiungo dei radio button:

```
<INPUT TYPE="radio" NAME="AnnoRadio"
  VALUE="2000">2000
<INPUT TYPE="radio" NAME="AnnoRadio"
  VALUE="1999">1999
```

- Per scegliere il secondo:

```
Document.Forma.AnnoRadio[1].checked =
  true;
```