Anno Accademico 2007-2008

Corso di Tecnologie Web

javascript

http://www-lia.deis.unibo.it/Courses/TecnologieWeb0708/

Javascript: Client Side Script

> Uno Script Client Side è un programma contenuto in un documento HTML

> Il supporto del linguaggio HTML è indipendente dal linguaggio di scripting

> Uno Scipt Client Side offre numerose possibilità di estendere il documento:

- modifica dinamica del documento in fase di caricamento
- modifica dinamica e elaborazione dei contenuti di una form
- esecuzione di operazioni al verificarsi di un evento

> In un documento possono essere presenti due tipi di script:

- script eseguiti una sola volta nella fase di caricamento
- script eseguiti al verificarsi di un evento

Javascript: HTML – Tag script

> Permette di includere uno script in un documento HTML

> Può essere usato più volte sia nella sezione <head> che nella sezione <body>

> Attributi:

- src = *uri*
  specifica l'indirzzo di uno script in un file esterno
- type = *content-type*
  specifica il linguaggio di scripting utilizzato
  (sostituisce language che è deprecato)

> E' possibile specificare un contenuto per browser che non supportano lo scripting con il tag <noscript>

Javascript

> Linguaggio di scripting a oggetti, cross-platform di Netscape

> Possiede un core di oggetti, funzioni, operatori e strutture di controllo

> Client-side Javascript estende il core con oggetti per il controllo del browser e del DOM (Document Object Model)

> Anche se si assomigliano Javascript è molto diverso da Java

JavaScript	**Java (Applet)**
Interpreted (not compiled) by client.	Compiled bytecodes downloaded from server, executed on client.
Object-oriented. No distinction between types of objects. Inheritance is through the prototype mechanism, and properties and methods can be added to any object dynamically.	Class-based. Objects are divided into classes and instances with all inheritance through the class hierarchy. Classes and instances cannot have properties or methods added dynamically.
Code integrated with, and embedded in, HTML.	Applets distinct from HTML (accessed from HTML pages).
Variable data types not declared (dynamic typing).	Variable data types must be declared (static typing).
Cannot automatically write to hard disk.	Cannot automatically write to hard disk.

Javascript

> Vediamo un esempio di ereditarietà nei modelli class based e prototype based

JavaScript	Java

function Employee () {
this.name = "";
this.dept = "general";
}
	public class Employee {
	public String name;
	public String dept;
	public Employee () {
	this.name = "";
	this.dept = "general";
	}
	}

function Manager () { this.reports = []; }
Manager.prototype = new Employee;
 	
function WorkerBee () {
this.projects = [];
}
WorkerBee.prototype = new Employee;
	public class Manager extends Employee
	{ public Employee[] reports;
	public Manager () {
	this.reports = new Employee[0];
	}
	}
	……

Javascript: Valori e Variabili

> Javascript riconosce i seguenti tipi di valori:

- Numerici (interi o floating point)
- Booleani (true or false)
- Stringhe
- null
- undefined

> Javascript è un linguaggio a tipizzazione dinamica:

```
var answer = 42;
answer = "Thanks for all the fish... ";
```

> Variabili:

- Il nome deve iniziare con una lettera o con _
- Javascript è *case sensitive*
- Assegnamento: x=42; var x=42;
- Una variabile non assegnata è *undefined*
- Scope: *global,local*

Javascript: Operatori di Assegnamento

Shorthand operator	Meaning
x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y
x <<= y	x = x << y
x >>= y	x = x >> y
x >>>= y	x = x >>> y
x &= y	x = x & y
x ^= y	x = x ^ y
x |= y	x = x | y

Javascript: Operatori di Confronto

var1=3; var2=4;

Operator	Description	Ex. returning true
Equal (==)	Returns true if the operands are equal. If the two operands are not of the same type, JavaScript attempts to convert the operands to an appropriate type for the comparison.	3 == var1 "3" == var1 3 == '3'
Not equal (!=)	Returns true if the operands are not equal. If the two operands are not of the same type, JavaScript attempts to convert the operands to an appropriate type for the comparison.	var1 != 4 var2 != "3"
Strict equal (===)	Returns true if the operands are equal and of the same type.	3 === var1
Strict not equal (!==)	Returns true if the operands are not equal and/or not of the same type.	var1 !== "3" 3 !== '3'
Greater than (>)	Returns true if the left operand is greater than the right operand.	var2 > var1
Greater than or equal (>=)	Returns true if the left operand is greater than or equal to the right operand.	var2 >= var1 var1 >= 3
Less than (<)	Returns true if the left operand is less than the right operand.	var1 < var2
Less than or equal (<=)	Returns true if the left operand is less than or equal to the right operand.	var1 <= var2 var2 <= 5

Javascript: Operatori Aritmetici

Operator	Description	Example
% (Modulus)	Binary operator. Returns the integer remainder of dividing the two operands.	12 % 5 returns 2.
++ (Increment)	Unary operator. Adds one to its operand. If used as a prefix operator (++x), returns the value of its operand after adding one; if used as a postfix operator (x++), returns the value of its operand before adding one.	If x is 3, then ++x sets x to 4 and returns 4, whereas x ++ sets x to 4 and returns 3.
-- (Decrement)	Unary operator. Subtracts one to its operand. The return value is analogous to that for the increment operator.	If x is 3, then --x sets x to 2 and returns 2, whereas x ++ sets x to 2 and returns 3.
- (Unary negation)	Unary operator. Returns the negation	

Javascript: Operatori bit a bit

Operator	Usage	Description
Bitwise AND	*a* & *b*	Returns a one in each bit position for which the corresponding bits of both operands are ones.
Bitwise OR	*a* | *b*	Returns a one in each bit position for which the corresponding bits of either or both operands are ones.
Bitwise XOR	*a* ^ *b*	Returns a one in each bit position for which the corresponding bits of either but not both operands are ones.
Bitwise NOT	~ *a*	Inverts the bits of its operand.
Left shift	*a* << *b*	Shifts a in binary representation b bits to left, shifting in zeros from the right.
Sign-propagating right shift	*a* >> *b*	Shifts a in binary representation b bits to right, discarding bits shifted off.
Zero-fill right shift	*a* >>> *b*	Shifts a in binary representation b bits to the right, discarding bits shifted off, and shifting in zeros from the left.

Javascript: Operatori logici

&&	expr1 && expr2	(Logical AND) Returns expr1 if it can be converted to false; otherwise, returns expr2. Thus, when used with Boolean values, && returns true if both operands are true; otherwise, returns false.
||	Expr1 || expr2	(Logical OR) Returns expr1 if it can be converted to true; otherwise, returns expr2. Thus, when used with Boolean values, || returns true if either operand is true; if both are false, returns false.
!	!expr	(Logical NOT) Returns false if its single operand can be converted to true; otherwise, returns true.

Javascript: Operatori Speciali

- new: crea un'istanza di un oggetto
- this: referenzia l'oggetto corrente
- typeof: ritorna il tipo di un oggetto
- void: valuta una espressione senza tornare alcun valore

Javascript: Operatori - Precedenza

Operator type	Individual operators
comma	,
assignment	= += -= *= /= %= <<= >>= >>>= &= ^= |=
conditional	?:
logical-or	||
logical-and	&&
bitwise-or	|
bitwise-xor	^
bitwise-and	&
equality	== !=
relational	< <= > >=
bitwise shift	<< >> >>>
addition/subtraction	+ -
multiply/divide	* / %
negation/increment	! ~ - + ++ -- typeof void delete
call	()
create instance	new
member	. []

Javascript: Istruzioni Condizionali

> If

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

> Condition

```
 variablename=(condition)?
value1:value2
```

> Switch:

```
switch (expression) {
   case label :
      statement;
      break;
   case label :
      statement;
      break;
   ...
   default :
statement;
}
```

Javascript: Loop

> For

```
for ([initialExpr];[condition];[incrementExpr]){
    statements
}
```

> Do

```
do {
    statement
} while (condition)
```

> While

```
while (condition) {
    statements
}
```

Javascript: Functions

> Definizione di Funzione (parametri passati sempre per valore)

```
function square(number) {
        return number * number;
}
```

> Chiamata di Funzione

```
square(5);
```

> E' possibile realizzare funzioni con numero variabile di parametri

> E' possibile definire Array multidimensionali

Javascript: Funzioni predefinite

> `eval(expr)`

valuta la stringa expr (espressione,istruzioni)

> `isFinite(number)`

number=NaN,+inf,-inf → false

> `isNaN(testValue)`

testValue = NaN → true

> `parseInt(str [, radix])`

converte la string str in un intero in base radix

> `parseFloat(str)`

converte la string str in un floating point

Javascript: Oggetti predefiniti (Core)

- Array
- Boolean
- Date
- Math
- Number
- String
- ……..

Javascript: Oggetto Array

- Creazione di un Array

```
arrayObjectName = new Array(element0,
element1, ..., elementN)
arrayObjectName = new Array(arrayLength)
```

- Popolamento di un Array

```
emp[1] = "Casey Jones"
emp[2] = "Phil Lesh"
emp[3] = "August West"
myArray = new Array("Hello", myVar, 3.14159)
```

- Riferimento ad un elemento

```
myArray[0]
```

Javascript: Gestione degli Eventi

> Le applicazioni Javascript sono principalmente basate sugli eventi

> Gli eventi si verificano come risultato di una azione dell'utente

> Per attivare uno script allo scatenarsi di un evento è necessario definire un gestore dell'evento (Event Handler)

Javascript: Eventi

Event	Applies to	Occurs when	Event handler
Abort	images	User aborts the loading of an image (for example by clicking a link or clicking the Stop button)	onAbort
Blur	windows and all form elements	User removes input focus from window or form element	onBlur
Change	text fields, textareas, select lists	User changes value of element	onChange
Click	buttons, radio buttons, checkboxes, submit buttons, reset buttons, links	User clicks form element or link	onClick
DragDrop	windows	User drops an object onto the browser window, such as dropping a file on the browser window	onDragDrop
Error	images, windows	The loading of a document or image causes an error	onError
Focus	windows and all form elements	User gives input focus to window or form element	onFocus
KeyDown	documents, images, links, text areas	User depresses a key	onKeyDown
KeyPress	documents, images, links, text areas	User presses or holds down a key	onKeyPress
KeyUp	documents, images, links, text areas	User releases a key	onKeyUp

Javascript: Eventi

Event	Applies to	Occurs when	Event handler
Load	document body	User loads the page in the Navigator	onLoad
MouseDown	documents, buttons, links	User depresses a mouse button	onMouseDown
MouseMove	nothing by default	User moves the cursor	onMouseMove
MouseOut	areas, links	User moves cursor out of a client-side image map or link	onMouseOut
MouseOver	links	User moves cursor over a link	onMouseOver
MouseUp	documents, buttons, links	User releases a mouse button	onMouseUp
Move	windows	User or script moves a window	onMove
Reset	forms	User resets a form (clicks a Reset button)	onReset
Resize	windows	User or script resizes a window	onResize
Select	text fields, textareas	User selects form element's input field	onSelect
Submit	forms	User submits a form	onSubmit
Unload	document body	User exits the page	onUnload

Javascript: Event Handler

> E' possibile definire un event handler per un tag che supporta il relativo evento.

```
<tag eventHandler="JavaScript Code">
```

> Esempio:

```
<input type="button" value="Calculate" onClick="compute(this.form)"/>
```

> E' possibile inserire più istruzioni in sequenza, ma è meglio definire delle funzioni.

> E' necessario alternare i doppi apici con l'apice singolo

```
<input type="button" name="Button1" value="Open Sesame!"
        onClick="window.open('mydoc.html', 'newWin')">
```

Javascript: Event Handler

```
<head>
  <script type="text/javascript">
    <!--- Hide script from old browsers
     function compute(f) {
       if (confirm("Are you sure?"))
         f.result.value = eval(f.expr.value)
       else
         alert("Please come back again.")
     }
    // end hiding from old browsers -->
  </script>
</head>
<body>
  <form>
    Enter an expression:
    <input type="text" name="expr" size=15 >
    <input type="button" value="Calculate“
           onClick="compute(this.form)">
    <br/>
    Result:
    <input type="text" name="result" size=“15” >
  </form>
</body>
```

Javascript: Event Handler

```
<script type="text/javascript">
  function fun1() {
     ...
  }
  function fun2() {
     ...
  }
</script>


<form name="myForm">
  <input type="button" name="myButton“ onClick="fun1()">
</form>


<script>
  document.myForm.myButton.onclick=fun2;
</script>
```

Javascript: Validare una input form

- Riduce il carico delle applicazioni server side filtrando l'input
- Riduce il ritardo in caso di errori di inserimento dell'utente
- Semplifica le applicazioni server side
- Consente di introdurre dinamicità all'interfaccia web (utilizzando anche nuove metafore d'interazione: es. ajax)
- Generalmente si valida formalmente un documento in due momenti:
  - Durante l'inserimento utilizzando l'evento onChange
  - Al momento del submit della form

Javascript: Validare una input form

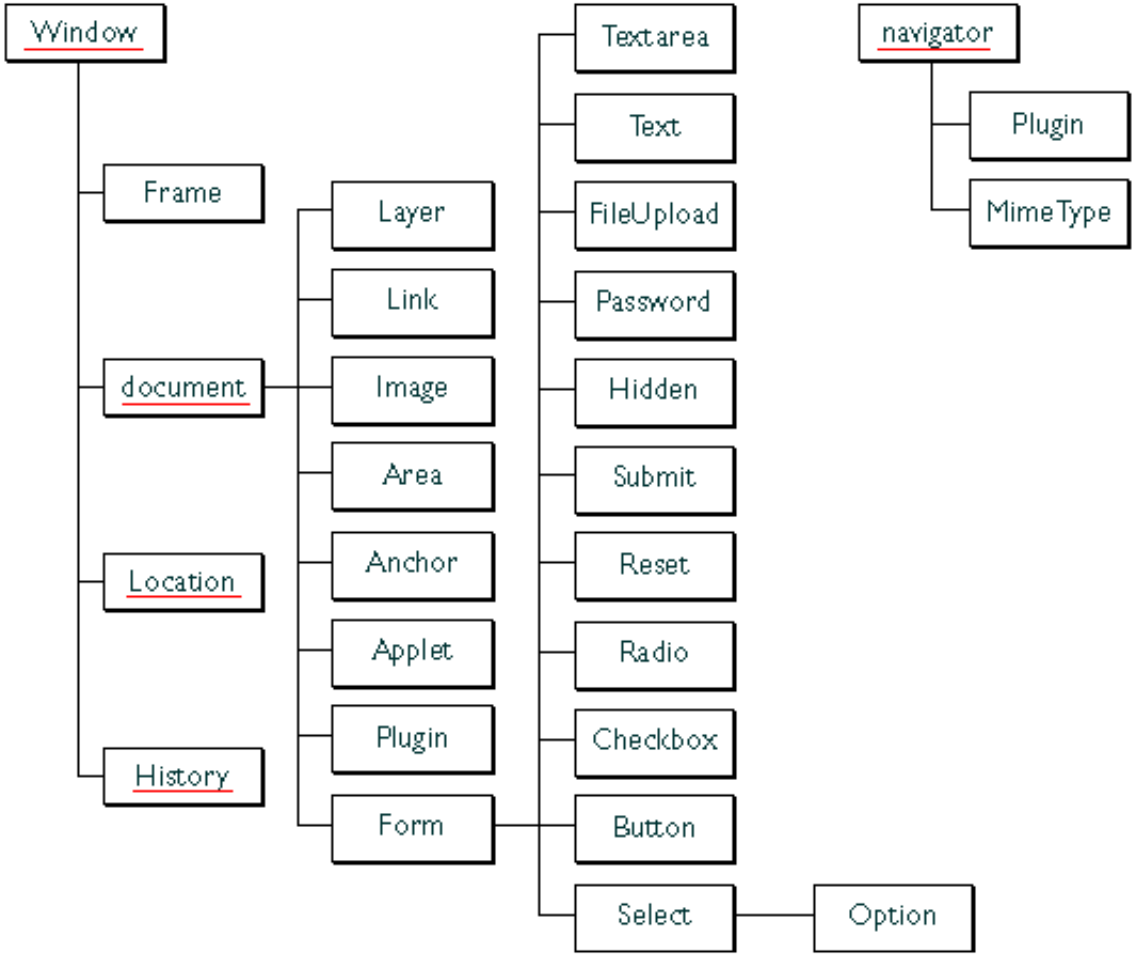
```
<head>
 <script type="text/javascript">
  function isaPosNum(s) {
   return (parseInt(s) > 0)
  }
  function qty_check(item, min, max) {
    var returnVal = false;
    if (!isaPosNum(item.value))
      alert("Please enter a positive number");
    else if (parseInt(item.value) < min)
      alert("Please enter a " + item.name + " greater than " + min);
    else if (parseInt(item.value) > max)
      alert("Please enter a " + item.name + " less than " + max);
    else
      returnVal = true;
    return returnVal;
  }


  function validateAndSubmit(theform) {
    if (qty_check(theform.quantity, 0, 999)) {
      alert("Order has been Submitted");
      return true;
    } else {
       alert("Sorry, Order Cannot Be Submitted!");
       return false;
    }
  }
 </script>
</head>
```

Javascript: Validare una input form

```
<body>
  <form name="widget_order" action="lwapp.html" method="post">
    How many widgets today?
    <input type="text" name="quantity" onchange="qty_check(this, 0, 999)">
    <br />
    <input type="submit" value="Enter Order“
            onclick="validateAndSubmit(this.form)">
  </form>
</body>
```

```
<form name="widget_order" action="lwapp.html" method="post“
        onSubmit="return qty_check(widget_order.quantity, 0, 999)">
  ...
  <input type="submit“ />
  ...
</form>
```

Javascript: Browser Objects



Javascript: Browser Objects

```
<head>
  <title>A Simple Document</title>
  <script type="text/javascript">
    function update(form) {
      alert("Form being updated")
    }
  </script>
</head>
<body>
  <form name="myform" action="foo.cgi" method="get" >
    Enter a value:
    <input type="text" name="text1" value="blahblah" size="20" >
    Check if you want:
    <input type="checkbox" name="Check1" checked="checked"
            onClick="update(this.form)">
    Option #1
    <br/>
    <input type="button" name="button1" value="Press Me"
            onClick="update(this.form)">
  </form>
</body>
```

Property	Value
document.title	"A Simple Document"
document.fgColor	#000000
document.bgColor	#ffffff
location.href	"http://www.royalairways.com/samples/simple.html"
history.length	7

Javascript: Browser Objects

```
<form name="myform" action="foo.cgi" method="get" >
   Enter a value:
   <input type="text" name="text1" value="blahblah" size="20" />
   Check if you want:
   <input type="checkbox" name="Check1" checked="checked"
            onClick="update(this.form)">
   Option #1
   <br/>
   <input type="button" name="button1" value="Press Me"
            onClick="update(this.form)" />
</form>
```

- document.myform is the form
- document.myform.Check1 is the checkbox
- document.myform.button1 is the button
- document.myform.action is http://www.royalairways.com/samples/mycgi.cgi, the URL to which the form is submitted.
- document.myform.method is "get," based on the value of the METHOD attribute.
- document.myform.length is 3, because there are three input elements in the form.
- document.myform.button1.value is "Press Me"
- document.myform.button1.name is "Button1"
- document.myform.text1.value is "blahblah"

Javascript: Document Object

- E' il contenitore di tutti gli elementi della pagina (Form, immagini, Link)
- Alcuni metodi:
  - write
  - writeln
- Alcune proprietà:
  - bgcolor
  - fgcolor
  - lastModified
  - cookie

Javascript: Form Object

> Un documento può contenere più oggetti form

> L'oggetto form può essere referenziato con il suo nome o col vettore forms[]

```
document.nomeForm
document.forms[n]
document.forms["nomeForm"]
```

> Gli elementi della form possono essere referenziati con il loro nome o col vettore elements[]

```
document.nomeForm.nomeElemento
document.forms[n].elements[m]
```

> Ogni elemento della form ha una proprietà che la riferisce

```
<form name="myForm">
 Form name:
 <input type="text" name="text1" value="test">
 <br/>
 <input name="button1" type="button" value="Show Form Name"
        onclick="this.form.text1.value=this.form.name">
</form>
```

Javascript: Form Object

- Alcune Proprietà:
  - action — riflette l'attributo action
  - elements — vettore contenente gli elementi della form
  - length — numero di elementi nella form
  - method — riflette l'attributo method
  - name — nome della form
  - target — riflette l'attributo target
- Alcuni Metodi:
  - reset — resetta la form
  - submit — esegue il submit
- Eventi:
  - onreset — evento scatenato quando la form viene resettata
  - onsubmit — evento scatenato quando viene eseguito il submit della form

Javascript: Form Object

```
<head>
  <title>Form object example</title>
  <script>
    function setCase (caseSpec) {
      if (caseSpec == "upper") {
        document.myForm.firstName.value=document.myForm.firstName.value.toUpperCase();
        document.myForm.lastName.value=document.myForm.lastName.value.toUpperCase();
      } else {
        document.myForm.firstName.value=document.myForm.firstName.value.toLowerCase();
        document.myForm.lastName.value=document.myForm.lastName.value.toLowerCase();
      }
    }
  </script>
</head>
<body>
  <form name="myForm">
    <b>First name:</b>
    <input type="text" name="firstName" size="20"/><br/>
    <b>Last name:</b>
    <input type="text" name="lastName" size="20"/>
    <p>
      <input type="button" value="Names to uppercase" name="upperButton"
onClick="setCase('upper')" />
      <input type="button" value="Names to lowercase" name="lowerButton"
onClick="setCase('lower')" />
    </p>
  </form>
```

Javascript: Form Object

```
…
<head>
   <title>Form object/onSubmit event handler example</title>
   <script>
     var dataOK=false;
     function checkData () {
       if (document.myForm.threeChar.value.length == 3)
         return true;
       else {
         alert("Enter exactly three characters. " +
               document.myForm.threeChar.value + " is not valid.");
         return false;
       }
     }
   </script>
</head>
<body>
   <form name="myForm" onSubmit="return checkData()">
     <b>Enter 3 characters:</b>
     <input type="text" name="threeChar" size= "3" />
     <p>
       <input type="submit" value="Done" name="submit1"
               onClick="document.myForm.threeChar.value=
                           document.myForm.threeChar.value.toUpperCase()" />
     </p>
   </form>
  …
```

Javascript: Form Object

```
…
<head>
   <title>Form object/submit method example</title>
   <script>
     var dataOK=false;
     function checkData () {
       if (document.myForm.threeChar.value.length == 3)
         document.myForm.submit();
        else {
          alert("Enter exactly three characters. " + document.myForm.threeChar.value +
                  " is not valid.");
          return false;
       }
     }
   </script>
</head>
<body>
   <form name="myForm" onSubmit="alert('Form is being submitted.')">
     <b>Enter 3 characters:</b>
     <input type="text" name="threeChar" size= "3" />
     <p>
       <input type="button" value="Done" name="button1" onClick="checkData()" />
     </p>
   </form>
  …
```

Javascript: Checkbox Object

> Alcune Proprietà:

- checked — booleano che definisce lo stato del checkbox (on/off)
- defaultchecked — riflette l'attributo checked
- form — specifica la form che lo contiene
- name — nome del checkbox
- value — riflette l'attributo value

> Alcuni Metodi:

- blur — toglie il focus dal checkbox
- focus — pone il focus sul checkbox
- click — simula il click del mouse sul checkbox

> Eventi:

- onblur — evento scatenato quando il checkbox perde il focus
- onfocus — evento scatenato quando il checkbox prende il focus
- onclick — evento scatenato quando l'utente clicca sul checkbox

Javascript: Checkbox Object

```
…
  <head>
    <title>Checkbox object example</title>
    <script>
      function convertField(field) {
        if (document.form1.convertUpper.checked)
          field.value = field.value.toUpperCase();
      }
      function convertAllFields() {
        document.form1.lastName.value = document.form1.lastName.value.toUpperCase();
        document.form1.firstName.value = document.form1.firstName.value.toUpperCase();
        document.form1.cityName.value = document.form1.cityName.value.toUpperCase();
      }
    </script>
  </head>
  <body>
    <form name="form1">
      <b>Last name:</b>
      <input type="text" name="lastName" size="20" onChange="convertField(this)" />
      <br/><b>First name:</b>
      <input type="text" name="firstName" size="20" onChange="convertField(this)" />
      <br/><b>City:</b>
      <input type="text" name="cityName" size="20" onChange="convertField(this)" />
      <p>
        <input type="checkBox" name="convertUpper"
                onClick="if (this.checked) {convertAllFields()}"> Convert fields to upper case
      </p>
    </form>
  …
```