Web-Based Information Systems

Fall 2006

CMPUT 410: JavaScript

Dr. Osmar R. Zaïane

University of Alberta

Publishing On the Web

- Writing HTML with a text editor allows to generate web pages. These pages are said static in the sense that they do not change.
- What if we want to personalize pages for particular visitors or events?
- What if we want to have actions on the page?
- What if the content of the page is from a database?
- Etc.

Course Content

• Introduction	• Perl & Cookies
• Internet and WWW	• SGML / XML
• Protocols	• CORBA & SOAP
• HTML and beyond	• Web Services
• Animation & WWW	• Search Engines
• CGI & HTML Forms	• Recommender Syst.
• **Javascript**	• Web Mining
• Databases & WWW	• Security Issues
• Dynamic Pages	• Selected Topics

Web-based Applications

Objectives

- Learn how JavaScript stores data, how a document is structured in JavaScript
- Learn event-based programming with JavaScript.
- Learn how JavaScript is event driven and how user actions are tracked
- See and analyze some concrete examples with JavaScript.

Content

I. JavaScript and the Details

- Variable identifiers and their types
- The notion of objets
- Arrays
- Control structures
  - Condition and selection
  - Iteration
- Procedures and functions

II. Event-Based Programming with JavaScript

- **What is an event?**
- What are the recognized events?
- Capturing events.

III. Practical Examples

- Data entry validation within a form;

Introduction to Variables

- A variable in Javascript has a type:
  - number (integer or non integer)
  - String
  - Boolean
  - Null
- JavaScript is not strongly typed.

Declaring Variables

The first time a variable is used it must be declared with the keyword **'var'.**

var identifier = value;

The identifier must start with a letter or underscore '_' and can have as many characters as necessary (letters, digits, underscore).

Javascript is sensitive to capital letters.
myvariable is different from *MyVariable* and x ≠ X

Type Conversion on the fly

- Because JavaScript is not strongly typed, it is possible to:
  - Change the type of a variable;
  - Do operations on variables of different types.
  - The major type, or default type, is string.

Variable Examples

```
<HTML>
<HEAD>
<TITLE>My First Java Script with variables</TITLE>
<script language="JavaScript">
 <!– hide script
    var myNumber=35;
    var myString="2004";
    var myOtherString="CMPUT410";
    var myAddition = myNumber+myNumber;
    var myConcatenation = myStyring + myOtherString;
    var myError = myNumber + myOtherString;
    var myCalculation = myNumber + myString;
    var myDream = myOtherString + myString;
 // end of hide -->
 </script>
</HEAD>
```


Variable Examples (con't)

```
<BODY>
<script language="JavaScript">
 <!-- hide script
        document.write("myAddition="+myAddition+"<BR>");
        document.write("myConcatenation="+myConcatenation+"<BR>");
        document.write("myError="+myError+"<BR>");
        document.write("myDream="+myDream+"<BR>");
        myError = myNumber * 3;
        document.write("myError="+myError+"<BR>");
        myNumber="Bye!";
        document.write("myNumber="+myNumber+"<BR>");
 // end of hide -->
 </script>
</BODY>
</HTML>
```

```
myAddition=70
myConcatenation=2004CMPUT410
myError=35CMPUT410
myDream= CMPUT4102004
myError=105
myNumber=Bye!
```


Content

JavaScript & Concept of Objects

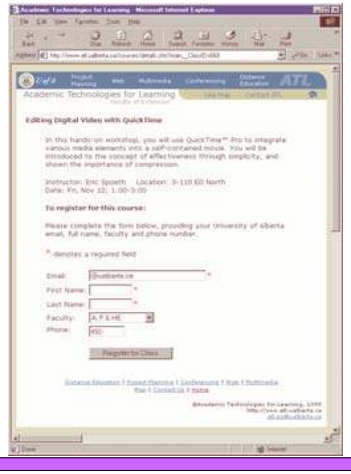
- JavaScript is not an object-oriented language.
- JavaScript is an object-based language.
- There are many pre-defined objects, but programmers can define their own objects.
- An object has attributes (specific properties) as well as methods (behaviour of objects).
- An attribute could be a value or recursively another object.

A Book is an Object

Title
Authors
Editors
Number of pages
Price
Set of Chapters
Set of figures and images
etc.

Each book has the same attributes with different values

What are the Objects, What are their Properties?



?

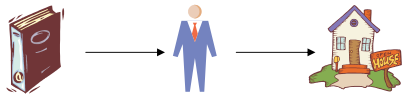
Access Object Properties

myObject.oneProperty

Object Name . Attributre Name

If the attribute is also an object, to access the property of the attribute's attribute:

myObject.oneProperty.aPropertyOfProperty
Ex: Book.Editor.Address



document.MyForm.Name.value

Access Object Methods

myObject.oneMethod(parameters)

Object Name . Method Name (parameters)

If there are no parameters:

myObject.oneMethod()

Ex: document.write("Hello!")

Predefined Object Classes

- There are many intrinsic pre-defined objects in JavaScript:

–Date	–Navigator
–String	–History
–Math	–Location
–Window	–Form
–Document	etc…

- These objects have their pre-defined attributes and methods.

Object Date

- The object Date needs to be instantiated with the keyword ***new***.

  var today= new date();

- The class Date doesn't have properties but the following methods:

•getDate()	•getYear()	
•getDay()	•setDate()	
•getHours()	•setHours()	etc…
•getMinutes()	•setMinutes()	
•getMonth()	•setMonth()	
•getSeconds()	•getSeconds()	
•getTime();	•setTime();	
•getTimezoneOffset()	•setYear()	

Example with Date

```
<HTML>
<HEAD>
<TITLE>My test with dates</TITLE><script language="JavaScript">
      var thisIsNow=new Date();
      var BirthDate = new Date(60,05,18);
</script></HEAD>
<BODY> <script language="JavaScript">
    document.write("Today we are the: "+thisIsNow+"<BR>");
    document.write("Alfred's birthdate is the "+ BirthDate +"<BR>");
    document.write("The date:"+ BirthDate.getDate()+ "/" +
                                (BirthDate.getMonth()+1) + "/" +
                                (BirthDate.getYear()+1900)+"<BR>");
     document.write("The time now is:" + thisIsNow.getHours() + ":" +
                                         thisIsNow.getMinutes() + ":" +
                                         thisIsNow.getSeconds()+"<BR>");
     thisIsNow.setYear(2010);
     document.write("The new date in the future is:<br>"+ thisIsNow);
  </script></BODY></HTML>
```


The Object String

- Where we define a string constant or a string variable, JavaScript creates an instance of an object String.
- The object String has one property, ***lenght***, and many methods:

•anchor()	astring.anchor(anchor)➔<A name="anchor">astring</A>
•big()	astring.big()➔<BIG>astring</BIG>
•blink()	astring.bink()➔<BLINK>astring</BLINK>
•bold()	astring.bold()➔<BOLD>astring</BOLD>
•fontcolor()	astring.fontcolor(#FF0000)➔<FONT color="#FF0000">astring</FONT>
•fontsize()	astring.fontsize(5)➔<FONT size=5>astring</FONT>
•italics();	astring.italics() ➔<I>astring</I>
•small()	astring.small()➔<SMALL>astring</SMALL>
•sub()	astring.sub() ➔_{astring}
•sup()	astring.sup() ➔^{astring}

The Object String (con't)

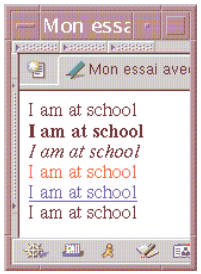
- Other methods for the String object:

•link()	
•toUpperCase()	astring.toUpperCase() converts into uppercase
•toLowerCase()	astring.toLowerCase() converts into lowercase
•substring()	astring.substring(3,5) substring from 3rd character to 5th.
•indexOf()	astring.indexOf(anOther) returns the position of anOther in astring.
•charAt()	astring.charAt(4) returns the 4th character.

The index in a string starts at 0.

Example with String Object

```
<HTML>
<HEAD>
<TITLE>My test with strings</TITLE>
<script language="JavaScript">
var myString=prompt("Give me a phrase","I am at school");
</script>
</HEAD><BODY>
<script language="JavaScript">
        document.write(myString+"<BR>");
        document.write(myString.bold()+"<BR>");
        document.write(myString.italics()+"<BR>");
        document.write(myString.fontcolor("red").blink()+"<BR>");
        document.write(myString.link("http://www.cnn.com")+"<BR>");
        document.write(myString +"<BR>");
 </script>
</BODY></HTML>
```



The Object Math

- The class Math contains common constants such as:
  Math.PI and **Math.E** which respectively return **3.1415926535897931** and **2.7182818284590451**
- Some useful methods:

•abs()	•log()
•acos()	•max()
•cos()	•min()
•asin()	•pow()
•sin()	•random()
•atan()	•round()
•tan();	•sqrt();
•exp()	•floor()

The Object Window

- JavaScript provides by default an object ***window*** representing the current window. This object is the root of the hierarchy describing the JavaScript objects.
- This object has many properties:

•defaultStatus	default message displayed on the status bar
•frames	set of frames displayed by the browser
•length	number of frames present in the parent window
•name	name of the window
•parent	parent window
•self	active window
•status	message displayed on the status bar
•top	top of the object hierarchy defined by the browser
•window	active window

Methods of the Window Object

- alert() (modal) window to display a message.
- confirm() (modal) window for selection.
- prompt() (modal) window to enter a value.
- clear() clears the window content.
- open() opens a new window.
- close() closes the current window.

To open a window:
window.open("URL","NameOfWindow","options");
Options are:

•menubar	•resizable	•toolbar
•status	•width	•location
•scrollbars	•height	•directories

Example with Windows

```
<HTML>
<HEAD>
<TITLE>My test with windows</TITLE>
</HEAD>
<BODY>
<A HREF="#" onMouseOver="window.status='Hi!';return true;">Hello!</A>
<BR>
<A HREF="test.html" TARGET="new">Open me</A>
<BR>
</BODY>
</HTML>
```


Opening a Window

```
<HTML>
<HEAD>
<TITLE>My other test with windows</TITLE>
</HEAD>
<BODY>
<a href="#" onClick="myWindow=window.open(
        'test.html','myTest','width=100,height=200,scrollbars,resizable');
        return true;">Open me</a><br>
<a href="#" onClick="myWindow.focus();return true;">Show me</a><br>
<a href="#" onClick="myWindow.blur(); return true;">Hide me </a><br>
<a href="#" onClick="myWindow.close();return true;">Close me</a><br>
</BODY>
</HTML>
```

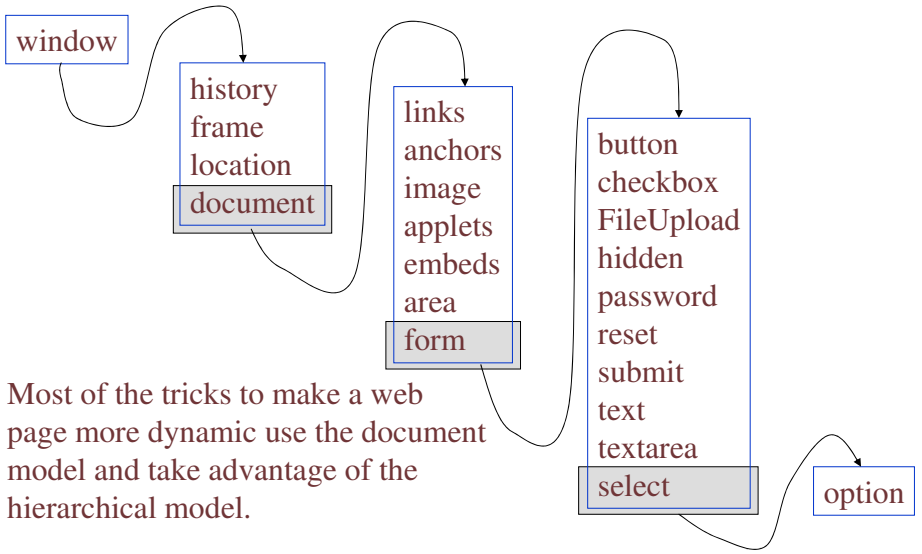

Example with Window Options

```
<HTML>
<HEAD>
<TITLE>Yet another test with windows</TITLE>
</HEAD>
<BODY>
menubar,status,scrollbars,resizable,location,directories,width,height
<BR>
<a href="#" onClick="option=prompt('What are your options:');
            myWindow=window.open('','myTest',option);
            return true;">Open me</a>
<BR>
<a href="#" onClick="myWindow.close();
        return true;">Close me</a>
<BR>
</BODY>
</HTML>
```

Document Object Model (DOM)

- Now that we know what a JavaScript object is and we know how to open a window, it is time to learn about the document object model.
- JavaScript includes predefined objects such as *window*. These objects have predefined properties and methods.
- An object property can also be an object.

Object Hierarchy



Most of the tricks to make a web page more dynamic use the document model and take advantage of the hierarchical model.

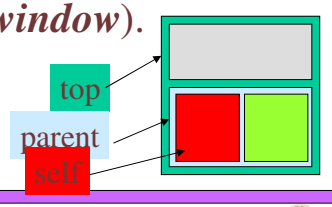
Other Examples of Predefined Objects

- History contains the list of all visited URL during the current session.

```
<HTML>
<HEAD><TITLE>My test with history</TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
  document.write("Number of URL="+window.history.length);
</SCRIPT><FORM>
<INPUT TYPE=button VALUE="Back"
        onClick="window.history.back();return true;">
<INPUT TYPE=button VALUE="Back 3 pages"
        onClick="window.history.go(-3);return true;">
</FORM>
</BODY></HTML>
```


The Frames

- The important thing about frames, is that each frame inside a window is also considered a window.
- This implies nested window.
- Thus, there is a window on top including all the others called ***top***, and each window has a parent called ***parent***. To reference the current window, we use ***self*** (interchangeable with ***window***).



JavaScript and Frames

```
<HTML><HEAD><TITLE>My test with Frames</TITLE>
<SCRIPT LANGUAGE="JavaScript">
var FrameEmpty = '<html><body bgcolor="#FFFFFF"></body></html>';
</SCRIPT>
</HEAD>
<FRAMESET rows="25%,*">
<FRAME SRC="EX16.HTM" name="controle">
<FRAME SRC="javascript:parent.FrameEmpty" name="target_frame">
</FRAMESET>
</HTML>
```

EX16.HTM

```
<HTML>
<HEAD><TITLE>Control Frame</TITLE></HEAD>
<BODY>
<a href="#" onClick="top.target_frame.document.writeln('Good Day!<br>');">Good Day!</a>
<BR><a href="#" onClick="top.target_frame.document.writeln('Hi!<br>');">Hi!</a>
</BODY></HTML>
```


JavaScript and Frames (con't)

```
<HTML><HEAD><TITLE>Another test with Frames</TITLE>
<SCRIPT LANGUAGE="JavaScript">
var FrameEmpty = '<html><body bgcolor="#FFFFFF"></body></html>';
</SCRIPT>
</HEAD>
<FRAMESET rows="25%,*">
<FRAME SRC="EX16b.HTM" name="control">
<FRAME SRC="javascript:parent.FrameEmpty" name="target_frame">
</FRAMESET>
</HTML>
```

EX16b.HTM

```
<HTML>
<HEAD><TITLE>Control Frame</TITLE></HEAD>
<BODY>
<a href="#" onClick="top.target_frame. document.bgColor='#00FF00';">Green</a>
<BR><a href="#" onClick="top.target. document.bgColor='yellow';">Yellow</a>
</BODY></HTML>
```


The Object Navigator

There are no methods associated but some attributes:

```
<html>
<head><title>Version of the  Navigator</title></head>
<body>
<h1> Let's see what browser it is</h1>
<hr>
<script language="javascript">
document.write("The version of this browser is: " + navigator.appVersion);
document.write ("<br>The Browser is: <B>" + navigator.appName + "</B>");
document.write("<br>Its code name is: " + navigator.appCodeName);
document.write ("<br>The OS is: " + navigator.userAgent);
</script>
</body></html>
```


Content

Arrays

- Variables can contain numbers, strings, and object references. There is also another type of information that JavaScript can manipulate: **Arrays**.

```
var products = new Array(“car”, “truck”,  “bike”);
```

product[0] ➔car
product[1] ➔truck
product[2] ➔bike
product.length ➔ 3

Simple Example with Arrays

```
<HTML>
<HEAD>
<TITLE>test with arrays</TITLE>
<SCRIPT LANGUAGE="JavaScript">
        var colours=new Array("red","blue","green","white");
</SCRIPT>
</HEAD>
<BODY>
<a href="#" onClick="document.bgColor=colours[0];return true;">Colour1</a>
<BR>
<a href="#" onClick="document.bgColor=colours[1];return true;">Colour2</a>
<BR>
<a href="#" onClick="document.bgColor=colours[2];return true;">Colour3</a>
<BR>
<a href="#" onClick="document.bgColor=colours[3];return true;">Colour4</a>
</BODY>
</HTML>
```


Another Test with Arrays

```
<HTML>
<HEAD>
<TITLE>Another test with arrays</TITLE>
<SCRIPT LANGUAGE="JavaScript">
        var processors=new Array("Intel PIII","AMD K7","Cirex");
</SCRIPT>
</HEAD><BODY>
<FORM NAME="myCPUform">
<INPUT TYPE="text" NAME="cpu" VALUE=""><BR>
<INPUT TYPE="button" VALUE="Intel"
   onClick="document.myCPUform. cpu.value=processors[0];return true;">
<INPUT TYPE="button" VALUE="AMD"
   onClick="document. myCPUform. cpu.value=processors[1];return true;">
<INPUT TYPE="button" VALUE="Cirex"
   onClick="document. myCPUform. cpu.value=processors[2];return true;">
</FORM>
</BODY></HTML>
```


Random Add

```
<HTML><HEAD>
<TITLE>Test with Arrays</TITLE>
<SCRIPT LANGUAGE="JavaScript">
    var pubs=new Array("car01.jpg","car02.jpg","car03.jpg","car04.jpg",
                       "car05.jpg","car06.jpg","car07.jpg");
    var numPub=pubs.length;
    var now = new Date();
    var x = now.getSeconds() % numPub;
</SCRIPT>
</HEAD><BODY>
<H1>Today’s Car</H1>
<SCRIPT LANGUAGE="JavaScript">
    document.write(“Car Number "+ x+"<br>");
    document.write("<IMG WIDTH=320 HEIGHT=240 SRC="+pubs[x]+">");
</SCRIPT></BODY></HTML>
```

DOM Revised

- Despite the fact that we didn't see all the object details in DOM, we have now a rough idea how it works and how it is used.
- After seeing the arrays, we know that a document is structured as follows:

document

links[]
anchors[]
images[]
area[]
form[]

```
document.image[2].src="…";
```

Change the 3rd document image.

Content

I. JavaScript and the Details

- Variable identifiers and their types
- The notion of objets
- Arrays
- Control structures
  - Condition and selection
  - Iteration
- Procedures and functions

II. Event-Based Programming with JavaScript

- **What is an event?**
- What are the recognized events?
- Capturing events.

III. Practical Examples

- Data entry validation within a form;

Condition and Selection

- There are many variations for this

if - then - else

Like many other languages, JavaScript provides control structures to execute instructions pending some conditions.

```
if (condition)  instruction;
else OtherInstruction;
```

```
if (condition)  instruction;
```

```
if (condition) {
        instruction1;
        instruction2;
        …
}
```

```
if (condition) {
        instruction1;
        instruction2;
        …
} else {
        instructionA;
        instructionB;
        …
}
```

Nested Selections

- We can also have many nested conditions

```
if (condition1)  {instructions1;}
else if (condition2) {instructions2;}
else if (condition3) {instructions3;}
else if (condition4) {instructions4;}
….
else {otherInstructions;}
```


Logic Operators

- Conjunction:
  - And **&&**
  - example: (price>=200 && member==true)
- Disjunction
  - Or ||
  - example: (age>26 || total==1000)
- Negation
  - Not !
  - example: (!finale && !(numbre==50))

Example with Condition

```
<HTML>
<HEAD><TITLE>Example with Condition</TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
        var month=prompt("What is your favorite colour?");
        if (pays.toUpperCase() == "BLUE")
                alert("Great! Me too.");
        else
                alert("That is ugly!");
</SCRIPT>
</BODY>
</HTML>
```


While Loop

Like many other languages, JavaScript provides control structures repetitions

```
while (condition)  instruction;
```

```
while (condition) {
        instruction1;
        instruction2;
        …
}
```

The execution of the instruction is repeted while the condition is true.

Example with Repetition

```
<HTML>
<HEAD><TITLE>quiz</TITLE>
<SCRIPT LANGUAGE="JavaScript">
        var now = new Date();
        var x = now.getSeconds();
        var answer=prompt("guess the seconds");
</SCRIPT></HEAD><BODY>
<SCRIPT LANGUAGE="JavaScript">
        while (answer!= x) {
                document.write(answer +"<br>");
                if (x< answer) alert("Too big");
                else alert("Too small");
                 answer =prompt(" guess the seconds ");
        }
        document.write("Bravo!");
</SCRIPT></BODY></HTML>
```


For Loop

```
for (init;condition;increment)  instruction;
```

```
for (init;condition;increment) {
        instruction1;
        instruction2;
        …
}
```

for (i=0;i<20;i++) document.write("*");

Example with *for* Loop

```
<html>
<head><title>Factorials</title></head>
<body>
<h1>Factorials from 1 to 9</h1>
<script language = "JavaScript">
<! - -  hide me
for (i=1, fact=1; i<10;i++, fact=fact *i){
        document.write (i + "! = " + fact);
        document.write ("<br>");
}
//  end hiding -->
</script>
</body>
</html>
```


Arrays and DOM

- As seen before, a document could have a set of images. These images could have a name: <IMG SRC="image.gif" **NAME="myName"** >
- The name of an image can be used to manipulate then image: document.myName.src="car.gif";
- With DOM, a document has an image array: document.images[0], document.images[1],…
- When can then manipulate the images by accessing this array: document.images[3].src="car.gif";

Example with images[]

```
<html>
<head><title>Images of the document</title>
</head><BODY>
<IMG SRC=dblace1.jpg><IMG SRC=dblace2.jpg>
<IMG SRC=dblace3.jpg><IMG SRC=dblace4.jpg>
<IMG SRC=dblace5.jpg>
<SCRIPT LANGUAGE="JavaScript">
var which=100;
while(which <0 | | which >4)
        which =prompt(Which image? (0 .. 4)','1');
window.document.images[which].src="car01.jpg";
</SCRIPT>
</BODY>
</html>
```

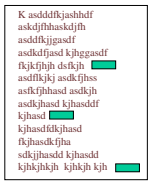

Content

Functions and Procedures

- Functions and procedures, also called subroutines, are fundamental in JavaScript programming.
- A subroutine is a set of instructions semantically linked.
- Grouping instructions in subroutines avoids rewriting the instructions. If there is an update, it suffices to make the change once in the sub-routine.

Example showing the use of subroutines with and without parameters

- Suppose we want to measure the reading speed of a user. We could put a button within the text that would display the time each time it is pressed.



The button could be:

```
<FORM>
<INPUT TYPE="button" VALUE="time" onClick="
        var theDate=new Date();
        var hour=theDate.getHours();
        var minutes=theDate.getMinutes();
        var secondes=theDate.getSeconds();
        var theTime=hour + ':' + minutes + ':' + secondes;
        alert(theTime);">
</FORM>
```


There is a Problem

- The previous code could be put in different places within the text.
- Notice that if the minutes or the secondes are less than 10, only one digit is displayed (0,1, 2, 3, …9) and not two (00, 01, 02,…09).
- Now we have to update the code wherever we incerted it.
- A better solution would be to write a subroutine that displays the time and call this same sub-routine with all the buttons in the text.

Sub-routine without Parameters

```
<SCRIPT LANGUAGE="JavaScript">
<!– hide me
    function announceTime() {
        var theDate=new Date();
        var hour=theDate.getHours();
        var minutes=theDate.getMinutes();
        var seconds=theDate.getSeconds();
        var theTime=hour + ':' + minutes + ':' + seconds;
        alert(theTime);
    }
// end hide -->
</SCRIPT>
```

```
<FORM>
<INPUT TYPE="button" VALUE="Time"
        onClick="announceTime();">
</FORM>
```


Updating the sub-routine

```
function announceTime() {
        var theDate=new Date();
        var hour=theDate.getHours();
        var minutes=theDate.getMinutes();
        if (minutes<10) minutes="0"+ minutes;
        var seconds=theDate.getSeconds();
        if (seconds<10) seconds="0"+ seconds;
        var theTime=heure + ':' + minutes + ':' + seconds;
        alert(theTime);
}
```

The change is done once and all buttons are updated.

Sub-routine with parameters

if (minutes<10) minutes="0"+ minutes;

if (secondes<10) seconds="0"+ seconds;

Similar operations ➔ we could create a new function to do this same operation.

Sub-routine with parameters and return value

arguments

```
function correct(theNumber) {
        var myValue;
        if (theNumber<10) myValue="0"+theNumber;
        else myValue=theNumber;
        return myValue;
}
```

Returned value

The updated sub-routine

```
function announceTime() {
        var theDate=new Date();
        var hour=theDate.getHours();
        var minutes=theDate.getMinutes();
        var corrMin = correct(minutes);
        var seconds=theDate.getSeconds();
        var corrSec = correct(seconds);
        var theTime=hour + ':' + corrMin + ':' + corrSec;
        alert(theTime);
}
```


Object Creation

- Functions are also used to define and create new objects.

```
function house(roomn, year, gar){
   this.rooms = roomn;
   this.year = year;
   this.aGarage = gar;
}
var myHouse = new house(8, 1990, true)
```

rooms
year
aGarage

Object - Array

•Each object is made of an array of values and properties.

•Thus, we could write:

myHouse[0] = 8;

myHouse[1] = 1990;

myHouse[2] = true;

Index by Property

- An array is indexed by an integer (0 to N).
- An array representing an object is also indexed by object attribute.
- Thus, we could write:

myHouse["rooms"] = 8;

myHouse["year"] = 1990;

myHouse["aGarage"] = true;

Dynamic Extension of Objects

- We can dynamically add attributes to an object.

yourHouse = new house(5, 1974, true);

yourHouse.closeOcean = "false";

yourHouse.windows = 18;

- The extension concerns only the instance to which the attributes were added. The other instances stay the same.

Attaching a method to an Object

- Like Object Oriented languages, JavaScript allows you to attach methods to objects that you create.

```
function house(roomn, year, gar){
   this.rooms = roomn;
   this.year = year;
   this.aGarage = gar;
   this.display = displayHouse;
}
```

Calling the Method

```
function displayHouse(){
   document.write("House has " + this.rooms + "
   rooms");
   if (this.aGarage)
      document.write(" and a garage.");
     document.write("Built in "+ this.year);
}
```

myHouse=new house(8,1990,true);
myHouse.display();

House has 8 rooms and a garage. Built in 1990

Summary Part I

- We saw the basics of JavaScript and we learned to write scripts and perhaps interpret scripts written by others.
- We explored the concept of object with JavaScript and covered some predefined objects such as String, Date, Math, Window, etc., as well as their properties and methods.
- We discussed the document object model (DOM).
- We saw how to create new objects and extend their properties and methods.

Content

I. JavaScript and the Details

- Variable identifiers and their types
- The notion of objets
- Arrays
- Control structures
  - Condition and selection
  - Iteration
- Procedures and functions

II. Event-Based Programming with JavaScript

- What is an event?
- What are the recognized events?
- Capturing events.

III. Practical Examples

- Data entry validation within a form;

Interactive Application

- For an interactive application one needs the possiblity to react to users actions.
- We need to provide the user with means to enter data, click on objects, select options, etc.
- The application needs to capture the users' actions and react to these actions.
- ➔ Identify events and answer them.

What is an Event?

- An event is a change in the environment due to some actions usually (but not always) made by the user.
- These actions are pertaining to the mouse and keyboard.
- A change in the document or the objects in the document constitute events.
- Example: moving or clicking the mouse, updating a value in an entry field, etc.

Content

What are the events recognised by JavaScript?

- Mouse click (left button).
- Mouse cursor passing over an object.
- The selection or de-selection of an entry field.
- The update of the entry field value.
- The submission of an entry form.
- The loading of a new document.
- The exit of the browser or document.

Events in JavaScript

•We already saw a sample of JavaScript events in the code examples used to illustrate previous concepts.
•JavaScript captures and manages 9 event types:

• onClick	The left button of the mouse is clicked when over a target.
• onMouseOver	The mouse cursor passes over a target.
• onBlur	The focus on a target is lost.
• onFocus	The target is activated.
• onSelect	The target is selected.
• onChange	The target content changed.
• onSubmit	The form is submitted (or being submitted).
• onLoad	The page is being loaded.
• onUnload	The page is being replaced or closed.

The Event Targets

The targets actually capture the events; these are objects from the document object model (DOM). Objects don't capture the same events.

• onClick	A HREF, BUTTON, CHECKBOX, RADIO, RESET, SUBMIT
• onMouseOver	A HREF
• onBlur	PASSWORD, SELECT, TEXT, TEXTAREA
• onFocus	PASSWORD, SELECT, TEXT, TEXTAREA
• onSelect	PASSWORD, TEXT, TEXTAREA
• onChange	PASSWORD, SELECT, TEXT, TEXTAREA
• onSubmit	FORM
• onLoad	BODY (window), IMAGE
• onUnload	BODY (window)

Other Events

- onError
- onAbort
- These events are used with window and image and come from page or image loading interruptions (manual interruption, abort, etc.).

How does it work?

- The browser intercepts the events (also called interruptions) and acts upon them.
- Action ➔ Event ➔Capture ➔Action
- The actions are associated to the targets by means of the HTML tags.
- <TAG onEvent="Action">
- Example: **<A href**="#" **onClick**="alert('Hello!');"**>**

Another Event: Timeout

- The window object has 2 specific methods to manage a countdown
- ***setTimeout()*** specifies a millisecond counter that is associated to an instruction. After a specified time interval, an interruption is produced and the instruction is evaluated.
- ***clearTimeout()*** cancels the countdown.

A living Clock

- We saw how to display the current time in a previous example.
- To have a clock, it suffices to contineously display the time at the same place.
- We have to call the function to display the time at regular intervals.
- Defining a timeout within the time display function would do the trick.

Example of a Timer

```
<html><head><title>Timer</title>
<SCRIPT LANGUAGE="JavaScript">
function myClock () {
        var now = new Date();
        var hour = now.getHours();
        var minutes = now.getMinutes();
        var seconds = now.getSeconds();
        var timeResult = "" + hour+ ((minutes < 10)?":0":":")
        + minutes + ((seconds < 10)?":0":":") + seconds;
        return timeResult;
}
function WhatTime() {
  var theTime=myClock();
  document.forms[0].clicking.value=theTime;
  setTimeout('WhatTime()',1000);
}
</SCRIPT></head>
```

```
<BODY onLoad="WhatTime()">
<FORM>
<INPUT TYPE=TEXT SIZE=8
        NAME="clicking">
</FORM>
</BODY></html>
```


Slides with setTimeout()

Continuously superposing images by calling the subroutine that displays an image at regular intervals.

```
<html>
<head><title>Countdown images</title>
<SCRIPT LANGUAGE="JavaScript">
var slides=new Array("jump1","jump2","jump3","jump4",
                     "jump5","jump6","jump7","jump8");
var number=slides.length;
var num=0;
var timer=0;
```


Slidess (con't)

```
function forward() {
        stop();
        num = (num+1) %number;
        window.document.images[0].src=slides[num] + ".jpg";
        window.document.forms[0].image.value=num+1;
}

function backward() {
        stop();
        if (num==0) num=number-1; else num--;
        document.images[0].src=slides[num] + ".jpg";
        window.document.forms[0].image.value=num+1;
}
```

Slidess (con't)

```
function next() {
        forward();
        timer=setTimeout("next()",500);
}

function stop() {
        clearTimeout(timer);
}

function restart() {
        timer=setTimeout("next()",500);
}
```


Slides (con't)

```
</SCRIPT>
</head>
<BODY>
<CENTER>
<IMG SRC="jump1.jpg">
<FORM>
<INPUT TYPE=BUTTON value="&lt;&lt;" onClick='backward();'>
<INPUT TYPE=BUTTON value="Stop" onClick='stop();'>
<INPUT TYPE=BUTTON value="&gt;&gt;" onClick='forward();'>
<INPUT TYPE=BUTTON value="Show" onClick='restart();'>
Image:<INPUT TYPE="TEXT" SIZE=1 NAME="image">
</FORM>
</CENTER>
</BODY>
</html>
```


Content

I. JavaScript and the Details

- Variable identifiers and their types
- The notion of objets
- Arrays
- Control structures
  - Condition and selection
  - Iteration
- Procedures and functions

II. Event-Based Programming with JavaScript

- What is an event?
- What are the recognized events?
- Capturing events.

III. Practical Examples

- Data entry validation within a form;

Order of Execution

- In traditional programming, the order of execution is dictated by the order of instrctions in the code. The execution is instruction by instruction.
- With event-based programming, the order of execution depends upon the events.
- When an event is intercepted, the corresponding instructions are executed.

Order of Execution (con't)

- Functions should be defined before they are invoked.
- **Warning**: An event could happen before the whole document is loaded. The user can indeed use the mouse while only part of the page is loaded and displayed.
- Before invoking a function we must make sure it is defined.
- This is one reason why it is better to put the function definitions in <HEAD>

Associating Functions with Events

- After defining a function, one must associate this function to the events we want to capture.
- We use the keywords for these events (event handlers) that we saw previously.

<BODY onLoad="JavaScript instructions">
<FRAMESET onLoad="JavaScript instructions">
window.onLoad=FonctionReference;

Example of Association

- A form **<FORM METHOD=...>...</FORM>** allows data entry. The browser submits these data to the server when the submit button is pressed.
- Adding onSubmit="**return verifyForm(this)**" to the tab <FORM> would call and execute the function verifyForm when the browser attempts to submit the data, and thus allows to intercept the data and validate it before it is actually sent.

Events for JavaScript Objects

- We saw the list of events intercepted by JavaScript.
- Let's see, object by object, the most common events and the HTML syntax.
- We will emphasise the events associated with form objects.

Common Events and *window*

- onLoad invoked when the page is loaded.
- onUnload invoked when the page is left.
- HTML Syntax
  - **<BODY**
    [BACKGROUND= "imageURL"] [BGCOLOR = "color"]
    [TEXT = "color"] [LINK = "color"] [ALINK = "color"] [VLINK = "color"]
    [onLoad = "handler"] [onUnload = "handler"] >
  - **<FRAMESET**
    [ROWS="lines"] [COLS="columns"]
    [onLoad="handler"] [onUnload="handler"]>

Common Events and *link*

- onClick invoked when a hyperlink is clicked.
- onMouseOver invoked when the mouse is over the link.
- HTML Syntax
  - **<A**
    HREF = "URL"
    [TARGET = "target_window"]
    [onClick = "handler"]
    [onMouseOver = "handler"] >

Common Events and *form*

- onReset invoked before resetting.
- onSubmit invoked before submitting.
- HTML Syntax
  - **<FORM**
    [NAME = "name"]
    [TARGET = "target_window"]
    [ACTION = "URL"]
    [METHOD = GET/POST]
    [ENCTYPE = "encryption"]
    [onReset = "handler"]
    [onSubmit = "handler"] >

Common Events and *button*

- onClick invoked when the button is pressed.
- HTML Syntax
  - **<INPUT**
    TYPE = button
    VALUE= "string"
    [NAME = "name"]
    [onClick = "handler"] >

Attributes

name
form
type
value

Common Events and *checkbox*

- onClick invoked when the checkbox is clicked.
- HTML Syntax
  - **<INPUT**
    TYPE = checkbox
    [VALUE= "string"]
    [NAME = "name"]
    [CHECKED]
    [onClick = "handler"] **>**

Attributes

name
form
checked
defaultChecked
type
value

Common Events and *radio button*

- onClick invoked when the radio button is clicked.
- HTML Syntax
  - **<INPUT**
    TYPE = radio
    [VALUE= "string"]
    [NAME = "name"]
    [CHECKED]
    [onClick = "handler"] **>**

Attributes

name
form
checked
defaultChecked
type
value

Common Events and *select*

- onChange invoked when a change occurs.
- onBlur invoked when the select looses focus.
- onFocus invoked when the select is activated.
- HTML Syntax
  - **<SELECT**
    NAME = "name"
    [SIZE= size]
    [MULTIPLE]
    [onChange = "handler"]
    [onBlur = "handler"]
    [onFocus = "handler"] **>**

Attributes

name
form
length
options
selectedIndex
type

Common Events and *text*

- onChange invoked when there is a change
- onBlur invoked when the text looses focus
- onFocus invoked when the text field is activated.
- HTML Syntax
  - **<INPUT**
    TYPE=text
    [NAME = "name"]
    [VALUE= "default"]
    [SIZE=size] [MAXLENGTH=maxsize]
    [onChange = "handler"]
    [onBlur = "handler"]
    [onFocus = "handler"] **>**

Attributes

name
form
value
defaultValue
type

Common Events and *textarea*

- onChange invoked when there is a change.
- onBlur invoked when the area looses focus.
- onFocus invoked when the area is activated
- HTML Syntax

```
•<TEXTAREA
    [NAME = "name"]
    [ROWS= "lines"] [COLUMNS="columns"]
    [WRAP=OFF/VIRTUAL/PHYSICAL]
    [onChange = "handler"]
    [onBlur = "handler"]
    [onFocus = "handler"]      >
```

Attributes

name
form
value
defaultValue
type

Common Events and *submit*

- onClick invoked the button is clicked.
- HTML Syntax

```
•<INPUT
    TYPE = submit
    [VALUE= "string"]
    [NAME = "name"]
    [onClick = "handler"]      >
```

Attributes

name
form
type
value

Common Events and *reset*

- onClick invoked the button is clicked.
- HTML Syntax

```
•<INPUT
    TYPE = reset
    [VALUE= "string"]
    [NAME = "name"]
    [onClick = "handler"]      >
```

Attributes

name
form
type
value

Common Events and *image*

- onClick invoked when the image is clicked.
- HTML Syntax

```
•<INPUT
    TYPE = image
    SRC= "URL"
    [NAME = "name"]
    [onClick = "handler"]      >
```

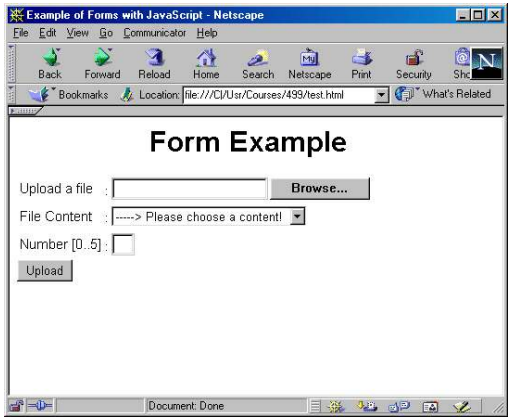
Attributes

name
form
type
src

Summary Part II

- We saw what is an event in JavaScript.
- We saw how events are intercepted.
- We enumerated the known events.
- We saw the form elements and the associated events.
- We saw how to program with events and how the execution is done.
- We illustrated the concepts with simple examples.

Content




```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<HTML>
<HEAD>
        <title>Example of Forms with JavaScript</title>
```



```
<SCRIPT  TYPE="text/javascript" LANGUAGE="JavaScript">
function checkform()
{
        if(document.myform.variable1.value=="") {
                alert("Please enter the full path for your file! ");
                document.myform.variable1.focus();
                return(false);
        }
        if(document.myform.variable2.selectedIndex==0) {
                alert("Please choose a content! ");
                document.myform.variable2.focus();
                return(false);
        }
        if(document.myform.variable3.value>5){
                alert("Please enter a correct number! ");
                document.myform.variable3.focus();
                return(false);
        }
}
</SCRIPT>
```

```
</HEAD>
<BODY bgcolor="#ffffff">
<center>
        <h1><font face="arial,helvetica"> Form Example</font></h1>
</center>
<FORM name="myform"
        action="/cgi-bin/mycgi.pl"
        method="post"
        enctype="multipart/form-data"
        onSubmit="return checkform()">
<table border="0">
<tr>
          <td><font face="arial,helvetica">  Upload a file</font></td>
          <td>: <input type="file" name= "variable1" size=20 > </td>
</tr>
```



```
<tr>
        <td><font face="arial,helvetica">  File Content</font></td>
        <td>: <select name="variable2">
                <option value="">-----> Please choose a content!
                <option value="1"> Resume
                <option value="2"> Application
                <option value="3"> Complaint
                <option value="4"> Other
                </select></td>
</tr>
<tr>
        <td><font face="arial,helvetica"> Number [0..5]</font></td>
        <td>: <input type="text" name= "variable3" size=2 > </td>
</tr>
</table>
<input type=submit value="Upload">
</FORM>
</BODY>
</HTML>
```


Entering an e-mail Address

```
<html>
<head><title>Validate my e-mail</title>
<SCRIPT LANGUAGE="JavaScript">

 •••

</script>
<BODY onLoad="emailReset();">
<FORM name="myAddress" onsubmit="return verifyEmail();">
E-Mail: <input type="text" name="email" size="25"
        onchange="this.value=checkEmail();"><br>
<input type="submit" value="Submit" name="B1">
<input type="reset" value="Init" name="B2" onclick="emailReset();">
</FORM>
</BODY>
</html>
```


e-mail Validation

```
function checkEmail(){
 var theAddress=document.myAddress.email.value;
 var newAddress = "";
 for (k = 0; k < theAddress.length; k++){
  ch = theAddress.substring(k, k+1);
  if ((ch >= "A" && ch <= "Z") || (ch >= "a" && ch <= "z") || (ch == "@") ||
      (ch == "[") || (ch == "]") || (ch == ".") || (ch == "_") || (ch == "-") ||
      (ch >= "0" && ch <= "9"))
       newAddress += ch;
 }
 if (theAddress!= newAddress) {
  if (confirm("You entered spaces or invalid characters.\n\n
     Click OK to fix this.\n\nClick CANCEL to keep unchanged."))
   return newAddress;
  return theAddress;
 }
 return theAddress;
}
```

e-mail Validation (con't)

```
function verifyEmail(){
  var theAddress = document.myAddress.email.value
  if (document.myAddress.email.value == "") {
    alert("Please enter your e-mail.")
    document.myAddress.email.focus();
    return false;
  }
 theAddress =document.myAddress.email.value;
  b = theAddress.substring(0,1)
  if (b == '@') {
    alert("Check your e-mail. You should have a prefix before '@'\n\n
            Example: jha@somewhere.com")
    document.myAddress.email.select();
    document.myAddress.email.focus();
    return false;
  }
```


e-mail Validation (con't)

```
  if ((theAddress.indexOf("@") == -1) || (theAddress.indexOf(".") == -1)) {
     alert("Check your e-mail.  An address must include the characters '@' and '.'\n\n
             Example: jha@somewhere.com");
     document.myAddress.email.select();
     document.myAddress.email.focus();
     return false;
    }
    c = theAddress.indexOf("@")
    d = theAddress.indexOf(".");
    e = theAddress.substring(c,d);
    if (e.length < 2) {
     alert("Some domain has to exist between the characters \"@\" and \".\"")
     document.myAddress.email.select();
     document.myAddress.email.focus();
     return false;
    }
```


e-mail Validation (con't)

```
  b = theAddress.indexOf(".")
  theAddress = theAddress.substring(b, theAddress.length);
  if (theAddress.length <2) {
    alert("You have to have a suffix of at least 2 characters after the '.'")
    document.myAddress.email.select();
    document.myAddress.email.focus();
    return false;
  }
  alert("Thank you");
  return false;
}

function emailReset(){
   document.myAddress.email.value = "";
   document.myAddress.email.focus();
}
```


Another e-mail example

```
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!-- Begin
function checkEmail(myForm) {
  if (/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/.test(myForm.emailAddr.value)){
   return (true)
  }
  alert("Invalid E-mail Address! Please re-enter.")
  return (false)
}
//  End -->
</script>
</HEAD>
```

```
<BODY>
<form onSubmit="return checkEmail(this)">
E-mail Address:<br>
<input type="text" name="emailAddr">
<p>
<input type="submit" value="Submit">
<input type="reset" value="Reset">
</form>
</BODY></HTML>
```

Return True

What is "return true", "return false"?
This is what is used to actually allow, or stop the form from submitting, respectively. This is how JavaScript controls the submitting of a form. By default, a form will return true. (Submit the form).
This is a important point. For example, you can stop a link from completing upon clicking.

```
<a href="normal.htm" onclick="return
false">Click here, it won't work!</a>
```

By returning false, we prohibit the action from completing!

Some useful References

- JavaScript with Netscape
  http://developer.netscape.com/docs/manuals/index.html?content=javascript.html
- Jscript (Microsoft) http://msdn.microsoft.com/scripting/default.htm
- Builder.com http://www.builder.com
- Designing with JavaScript http://www.webcoder.com/
- Ask the JavaScript Pro http://www.inquiry.com/techtips/js_pro/
- WebMonkey http://hotwired.lycos.com/webmonkey/programming/javascript/
- Yahoo
  http://dir.yahoo.com/Computers_and_Internet/Programming_Languages/JavaScript/