JavaScript part 1

A scripting language based on objects

JavaScript - a scripting language

- JavaScript was designed to add interactivity to HTML/XHTML pages
- JavaScript is a scripting language; it consists of lines of executable computer code
- A JavaScript may be directly embedded into HTML/XHTML pages:

```
<script type = "text/javaScript">
 -- JavaScript script –
 </script>
```

- Or indirectly, as a file specified in the src attribute of `<script>`, as in

```
<script type = "text/javaScript"
        src = "myScript.js">
 </script>
```


JavaScript - a scripting language

- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- The JS statements are the same on the client side as on the server side.

Using JavaScript

- Use JavaScript to create dynamic XHTML pages that receive data from the user, transform the data, store the data etc.
- JavaScript works in all major browsers, such as Internet Explorer, Firefox, Netscape, and Opera.
- A JS application can communicate with applications written in JAVA and CORBA.

What can JavaScript do?

It enhance the dynamics and interactive features of your HTML/XHTML page by allowing you to

- **perform calculations,**
- **check forms,**
- **write interactive games,**
- **add special effects,**
- **customize graphics selections,**
- **create security passwords**
- **and more.**

Starting JavaScript

To insert a JavaScript script into a HTML page, we use the <script> tag.

Inside the <script> tag we use the "type=" attribute to define the scripting language.

```
<html>
 <body>
<script type="text/javascript">
...
</script>
</body>
</html>
```


Scripts are usually hidden from browsers that do not include JavaScript interpreters by putting them in special comments

```
<script type="text/javascript">
//<![CDATA[
…………………………………………………………..
//]]>
</script>
```


A JavaScript statement is a command to the browser.

JavaScript code (or just JavaScript) is a sequence of JavaScript statements. Each statement is executed by the browser in the sequence they appear in the XHTML document.

```
<script type="text/javascript">
//<![CDATA[
window.document.write('<p>This is paragraph 1.</p>');
//]]>
</script>
```

JavaScript in a HTML page

- **Scripts in the body section**

Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates content in the page.

(2010_js_example 0_0.html)

- **Scripts in the head section**

Scripts to be executed when they are called, or when an event is triggered.

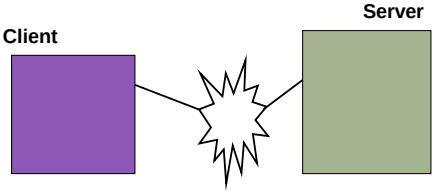
The script is loaded before anyone uses it. (2010_js_example 0_1.html)

- **External Scripts**

If you want to run the same JavaScript on several pages, without having to write the same script on every page, you can write a JavaScript script in an external file. Save the external JavaScript file with a .js file extension, and use reference to it in your .html file.

(2010_external_js.js and 2010_external_js.html)

Client/Server model



- **The client asks for an XHTML document.**
- **The server sends the document.**
- **The browser on the client side reads the XHTML document top-down and executes the XHTML and JS statements.**


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>2010_js_example_0_0</title>
</head>
<body style="color:black; font-family: arial;" link="#0000ff"
   vlink="#0000ff" alink="#0000ff">
<script type="text/javascript">
//<![CDATA[
window.document.write('<br /><h2 style="color:red;">The following
   message is written by javascript</h2>');
window.document.write('<p>This is paragraph 1.</p>');
//]]>
</script>
<hr />
<h2><font style="color:green;">This message is not written by
   javascript</font></h2>
<hr />
<script type="text/javascript">
//<![CDATA[
window.document.write('<br /><h2 style="color:red;">The following
   message is written by javascript</h2>');
window.document.write('<p>This is paragraph 2.</p>');
//]]>
</script>
</body>
</html>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
   Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
   transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Example 0_1</title>
<script type="text/javascript">
/*<![CDATA[*/
function example()
{
window.document.write('<br />Message written and background
   color changed by Javascript');
window.document.body.bgColor="yellow";
}
/*]]>*/
</script>
</head>
<body style="color:black; font-family: arial;" link="#0000ff"
   vlink="#0000ff" alink="#0000ff" onclick="example()">
<h3>This is the text written by html</h3>
<h5>The javascript message is not yet displayed.  Click on
   the page to execute the Javascript function</h5>
</body>
</html>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
   Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
   transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title> XHTML and Javascript external </title>
<script type="text/javascript"
   src="2010_external_js.js"></script>
</head>
<body style="color:black; font-family: arial;"
   link="#0000ff" vlink="#0000ff" alink="#0000ff"
   onclick="display_alert()">
Click on the page to execute the external .js file
<h3>The content of the js external file is:</h3>
<pre>
function display_alert()
{alert("I am an alert box!!\nGreetings CSE1550!!");}
</pre>
</body>
</html>
```


JavaScript comments

- Comments on a line
- Comments on many lines

```
<script type="text/javascript">
//This is a comment on a line
window.document.write("<h1>This is a header</h1>");
window.document.write("<p>This is a paragraph</p>");
/* This is a comment on
two lines */
window.document.write("<p>This is another
  paragraph</p>");
</script>
```


JavaScript variables

Variables are "containers" for storing information:

```
x=5;  length=66.10; X=6; Length=123;
```

JavaScript variables are used to hold values or expressions.

A variable can have a short name, like x, or a more describing name like length_x.

A JavaScript variable can also hold a text value like in `carname="Volvo"`.

Variable names are case sensitive (y and Y are two different variables)
Variable names must begin with a letter or the underscore character

A variable's value can change during the execution of a script. You can refer to a variable by its name to display or change its value.

```
x=7;
y=12;
X=x+y;// three variables x, y, X.
```


JavaScript variables

Variables (primitive type) in JavaScript: Number, String, Boolean, Undefined, or Null

Number, String, and Boolean have wrapper objects (*Number*, *String*, and *Boolean*). In the cases of Number and String, primitive values and objects are coerced back and forth so that primitive values can be treated essentially as if they were objects

All numeric values are stored in double-precision floating point.
String literals are delimited by either ' or "

Boolean values are `true` and `false`

The only Null value is `null`

The only Undefined value is `undefined`

An arithmetic operation that creates overflow returns `NaN` (Not a Number)

JavaScript strings

```
x="ABBA"
y="1zqerty23456"
```

`x.length`	Returns 4
`x.charAt(i)`	Returns the character at the specified index
`x.indexOf(":")`	Returns the position of the first found occurrence of a specified value in a string
`x.substr()`	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
`x.substring()`	Extracts the characters from a string, between two specified indices
`x.toLowerCase()`	Converts a string to lowercase letters
`x.toUpperCase()`	Converts a string to uppercase letters
`x.blink()`	Displays a blinking string
`x.sub()`	Displays a string as subscript text
`x.sup()`	Displays a string as superscript text

2010_String_Functions.html; 2010_Loops.html

JavaScript Arrays

An array may be considered as a group of variables having the same name. We may identify each variable from the group using an index.

```
var xarray = new Array(4);
xarry[0]=3;
xarray[1]=1;
xarray[2]="mac";
xarray[3]=12.3;  //not the same type of variable
```

or

```
var xarray = new Array(3, 1, "mac", 12.3);
```

or

```
var xarray = [3, 1, "mac", 12.3];
```

We will obtain the elements of an array using the name of the array and an index.

JavaScript operators

```
x=7;
y=12.123;
X=x+y;
x1="Mid-term test";
```

When you assign a text value to a variable, you use quotes around the value. We say that x1 is a string variable.

- Example of arithmetic operators: `+,-, /, *,++, --.`

The + operator can also be used to add string variables or text values together.

To add (concatenate) two or more string variables together, we use the + operator.

```
txt1="What is your ";
txt2="name?";
txt3=txt1+txt2;
```

If you need a space in a string variable you must include it in the string.

JavaScript array functions/properties

```
x=[3, 2, 1, 4]
y=["A", "B"]
```

`x.length`	→ 4
`x.sort()`	→ [1, 2, 3, 4]
`x.join(":")`	→ "3:2:1:4"
`x.reverse()`	→ [4, 3, 2, 1]
`x.concat(y)`	→[1, 2, 3, 4,"A", "B"]
`x.pop()`	Removes the last element of an array, and returns that element
`x.push("cc")`	Adds new elements to the end of an array, and returns the new length
`x.shift()`	Removes the first element of an array, and returns that element
`x.unshift("V")`	Adds new elements to the beginning of an array, and returns the new length
`x.toString()`	Converts an array to a string, and returns the result

2010_JS_Arrays.htm ; 2010_Array_Functions.html

JavaScript Functions (1)

A function definition has a header (reserved word function and the name of the function) and a body. Functions definitions must be placed in the HEAD of the HTML document.

```
function myfunction()
{.....body of the function}
```

A function may contain zero or more return statements.

The parameters in the definition of a function are called formal parameters.

```
function myfunction(x, y)
{.....}
```


JavaScript Functions (2)

The parameter values that appear in a call to a function are called actual parameters. When a function is called, the values of the actual parameters are copied into their corresponding formal parameters. *Pass-by-value*.

```
function myfunction(x, y, z, t)
{body of function}
. . . . .
myfunction(4,89.4,0.12,99);
```

There is no type or number checking of parameters.

```
myfunction(23.4, 670, 8);   //the value for t is undefined
myfunction(1, 2, 3, 4, 5, 6);
```


JavaScript functions (3)

A function is a reusable code-block that will be executed by an event, or or by a call to the function.

```
<html>
<head>
<script type="text/javascript">
function firstFunction()//no argument function
{window.document.bgColor = 'red';}
</script>
</head>
<body >
<form>
<input type="button" onclick="firstFunction()"
value="Call firstFunction">
</form>
<p>By pressing the button, firstFunction will be called
  and executed.</p>
</body>
</html>
```

2010_Example_Functions.html, 2010_external_2.txt

JavaScript functions (4)

- If you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.
- If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

2010_Example_FuctionWithArguments.html
2010_example_function_return.html
2010_computing_1.html

JavaScript events

List with events in JavaScript

Events are actions that can be detected by JavaScript.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input box in an HTML form
- Submitting an XHTML form
- A keystroke

OnMouseOver, OnMouseRigthClick, OnMouseOut, OnLoad, OnUnload.

Life objects versus programming objects

An object is a "thing" (data type) which has properties and methods.

Car

properties: wheels, windows, brakes

methods: runs on gas, pollutes, makes noise, one can drive it

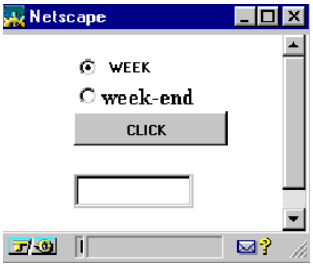
Book

properties: table of contents, author, title, language, nbr. of pages

methods: open it, read it, look for it in a library

JavaScript Objects and their hierarchy

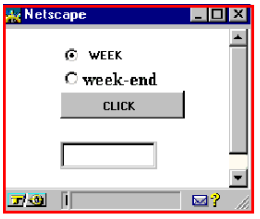
This is a Web page. JavaScript will divide this page in objects. You may access theses objects, their properties, and their methods.



JavaScript Objects and their hierarchy

The page is displayed in a Window.

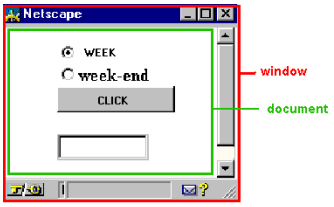
It is the object Window. The Window object has the window property that refers to itself.



window

JavaScript Objects and their hierarchy

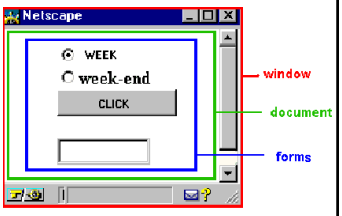
Every Window object contains a document property that refers to the Document object associated with the window.



window
document.

JavaScript Objects and their hierarchy

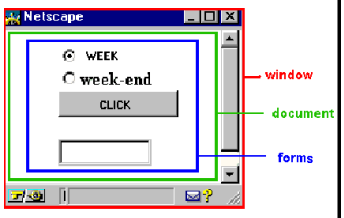
Every Document object has one or many properties named forms that contains Form objects.



window
document
forms[0]
forms[1]

JavaScript Objects and their hierarchy

The Document object has a method, write, which dynamically creates content, the parameter of the method write is a string, often concatenated from parts, some of which are variables

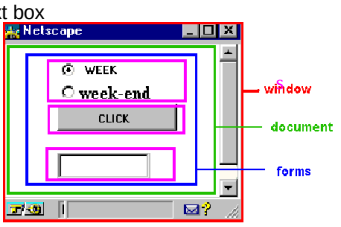


window.document. write()

JavaScript Objects and their hierarchy

There are two radio buttons, a plain button and a text box in the form

window
document
forms
radio[0]
radio[1]
button
text



JavaScript Objects and their hierarchy

To access an object you have to give the complete reference starting from outside.

Example:

the radio button week will be referenced as

window.document.forms[0].radio[0].

JavaScript Objects and their hierarchy

A window contains one document.

A document may contain zero, one or many forms (an array of forms).

A form contains one or many elements (radio buttons, text boxes, buttons, etc.)

JavaScript Object properties

A property is an attribute, a characteristic, an use or a quality of an object.

Example: the object book has the following properties: author, title, nbr. pages, etc.

Example: the object document has the bgcolor property.

To use a property you must use the name of the object, followed by a dot, followed by the name of the property.

```
window.document.bgColor
```


JavaScript Object properties

```
window.document
window.location
window.document.bgColor
window.document.cookie
window.document.forms[]
window.document.images[]
window.document.form.elements[i]
window.document.form.button_name
window.document.form.radio_name
```

JavaScript Objects methods

When you need to DO something, like open a window, write text to the screen, assign today's date to a variable, send the user back to the previous page, or display an alert box you are using a method of a specific object.

```
window.focus()   //window object methods
window.alert()
window.open()
window.close()
```


JavaScript Object methods (events)

```
window.document.open()
window.document.write()
window.document.close()
window.document.writeln()
window.document.form.onsubmitt()
window.document.form.reset()
window.document.form.button.onclick()
window.document.form.radio.onclick()
```

2010_js_Example_0_0.html
2010_js_Example_0_1.html
2010_External_js.html

Open a window

How to open a new window using JavaScript.

open(URL, windowName, windowFeatures)

In event handlers, you must specify window.open() instead of simply using open(). Due to the scoping of static objects in JavaScript, a call to open() without specifying an object name is equivalent to document.open().

The open method opens a new Web browser window on the client. If URL is an empty string, a new, empty window is created. If an URL is specified the document at the URL address will be displayed in the new window.

```
open('', 'messageWindow', 'directories=yes width=300,height=300')
```


windowFeatures

```
window.open('', 'messageWindow',
'directories=yes width=300,height=300')
```

directories

If yes, creates the standard browser directory buttons, such as What's New and What's Cool.

height

Specifies the height of the window in pixels.

menubar

If yes, creates the menu at the top of the window.

width

Specifies the width of the window in pixels.

windowFeatures

resizable

If yes, allows the user to resize the window.

scrollbars

If yes, creates horizontal and vertical scrollbars when the Document grows larger than the window dimensions.

status

If yes, creates the status bar at the bottom of the window.

2010-sesame.html

```
<body style="color:white; font-family: arial; background-color:black"
link="#0000ff" vlink="#0000ff" alink="#0000ff">
   <pre>
      *
     ***
    *****
     ***
      *
   </pre>

 I am the Ali Baba's Cavern!!

</body>
```


2010_Open-sesame.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title>
         Open sesame
      </title>
</head>
   <body style="color:black; font-family: arial;"
link="#0000ff" vlink="#0000ff" alink="#0000ff">
<form action="">
      <input type="button" name="button1" value="Open Sesame!"
      onClick="window.open ('2010_sesame.html',
'newWindow','scrollbars=yes, status=yes, width=300,
height=300')">
</form>
   </body>
</html>
```