JAVASCRIPT

-Les scripts peuvent être écrits.

-la déclaration d'une fonction javascript commence par « function »

-si la fonciton js return false le navigateur arrête l'exécution sinon il continue.

-document.locaton.href="page_name" permet de rediriger l'utilisateur vers une autre page

-pour inclure une nouvelle page à la page actuelle.

<jsp :include page="ajouterivlle.jsp" >

<jsp :param name=… value=.. />

</jsp :include>

It is a good idea to place scripts at the bottom of the <body> element.
This can improve page load, because script compilation can slow down the display.

-On peut mettre le code javascript dans un fichier extern et l'inclure en utilisant la balise :

<script src="myScript.js"></script>

A condition que ce script ne contiendra pas de balise <script>

JAVASCRIPT OUTPUT

- Writing into an alert box, using **window.alert()**.
- Writing into the HTML output using **document.write()**.
- Writing into an HTML element, using **innerHTML**.
- Writing into the browser console, using **console.log()**.
- -confirm(msg) : renvoie un boolean
- -alert(msg)
- -prompt(msg) : renvoie la valeur entrée par l'utilisateur

-Using document.write() after an HTML document is fully loaded, **will delete all existing HTML**

-*document.getElementById("element_id").innerHTML=* "Nouveau contenu" : permet de modifier le contenu de la balise qui l'id « element_id »

-on peut mettre les strings entre guillemets ou simple quotes, preferez les simples quotes si votre string contient des guillemets et vice versa .

-If a JavaScript statement does not fit on one line, the best place to break it, is after an operator

-en javascript, une variable non initialisé contient la valeur « **undefined** », le type de cette variable est aussi undefined.

-**undefined** et **null** ont la même valeur mais ne sont pas du même type.

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Javascript variables :

In JavaScript there are 5 different **data types** that can contain values:

string

number

boolean

object

function

There are 3 **types of objects**:

Object

Date

Array

And 2 data **types that cannot contain values**:

null

undefined

La même variable peut prendre différent type selon son contenu :

Var x=5 ;// number

x= *"aze" ; // string*

x={*"azer","qw"} ; // strings array*

x= {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"}; // objects array

objectName.methodName withtout () will return the function definition.

In JavaScript, scope is the set of variables, objects, and functions you have access to.

-affecting a value to undeclared variable makes it global.

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

.	Member	person.name
[]	Member	person["name"]

Methods are stored in properties as function definitions.

HTML EVENTS AND JAVASCRIPT RESPONSES :

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

See more events here

example with **this** keyword :

*<button **onclick**="**this**.innerHTML=Date()">The time is?</button>*

TYPES :

typeof x return the type of the objet, **it returns « object » if the variable is an array.**

It's better to use **instanceof,** because it returns « array » if the variable is an array.

Strings :

Name.Length to get its length

Infinity (also NEGATIVE_INFINITY and POSITIVE_INFINITY) and NaN are both values of the type Number. (the function isNan(value) test if the the value is a number or not)

charAt(position) ;

Dates :

-new Date() creates a date object with the current date and time.

-new Date(date string), creates a new date object from the specified date and time

ex : new Date("October 13, 2014 11:13:00")

-new Date(number), creates a new date object as zero time (1/1/1970 00 :00) plus the number in milliesecond.

-new Date(year, month, day, hours, minutes, seconds, milliseconds)

Arrays :

Arrays may contain objects,functions and even other arrays.

myArray[0] = Date.now;

myArray[1] = myFunction;

myArray[2] = myCars;

-array object : the valeus are written as name:value pairs

-to access an object properties we can choose one of the following choices :

objectName.propertyName or objectName["propertyName"]

-to access an object method : objectName.methodName()

x= {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"}; // objects array

- to access a such array object propertie : x.property_name
- all array methods and properties will produce incorrect results if used on an objects array

Arrays Utilities :

x.**length** returns the length of the array

x.**toString**() converts an array to a string of (comma separated) array values.

x.**join**(String separator) does the same thing except that it allow you to specify your own separator.

Popping items out of an array, or pushing items into an array :

x. **push**(value) adds « value » to the array « x » and returns the new length of the table.

also we can use x[x.length]=value to do the same thing

x. **pop**() **removes** the last element of the array (the one with the highest index) and **returns it**

sort() method sorts an array alphabeticallyto

to sort an array numerically use sort(comp_function())

x.concat(array…table) to concatenate two or more tables.

Note : Adding elements with high indexes can create undefined "holes" in an array

We can use « for » to loop through an array.

The Difference Between Arrays and Objects

In JavaScript, arrays use numbered indexes.

In JavaScript, objects use named indexes.

Regular expression :

In the following « reg » is either a litteral or a variable string

We use regular expressions in reg.search(String where_to_search) and reg.replace(String) methods.

Reg.test(String) returns true if any string matchs the pattern and false otherwise.

There is also reg.exec(String) that returns the string if found and null otherwise.

Regular expressions starts with a forward slash « / »

/i : makes the search case insensitive.

/g : Perform a global match (find all matches rather than stopping after the first match)

/m Perform multiline matching

Regular Expression Patterns

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |

Metacharacters are characters with a special meaning:

Metacharacter	Description
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning or at the end of a word
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers define quantities:

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*

Throw and Try to Catch :

try {

Block of code to try

}

catch(err) {

Block of code to handle errors

}

The throw statement allows you to create a custom error.

We can throw Strings and numbers

Debugging :

The console.log() Method to display logs to the browser's console (that can be opened with the f12 key)

The debugger keyword stops the execution of JavaScript, and calls (if available) the debugging function.

This has the same function as setting a breakpoint in the debugger.

Hoisting :

Hoisting is JavaScript's default behavior of moving all declarations to the top of the current scope (to the top of the current script or the current function).

That means that we can use a variable before declaring it.

JavaScript only hoists declarations, not initializations.

The "use strict" Directive

The "use strict" directive is new in JavaScript 1.8.5 (ECMAScript version 5).

It is not a statement, but a literal expression, ignored by earlier versions of JavaScript.

The purpose of "use strict" is to indicate that the code should be executed in "strict mode".

With strict mode, you can not, for example, use undeclared variables **AFTER THE LINE WHERE YOU USED THE « use strict » DIRECTIVE.**

HERE : http://www.w3schools.com/js/js_conventions.asp

Event :

addEventListener('click',myFunction(**e**),false)/**removeEventListener**(…same…) // e will contain the event, it's an optional parameter

Event.**type**

e.target.*property* (example id) // to get properties of the object that started the event.*

Conversion :

-parseInt() -parseFloat()

-Escape() // retourne la valeur hexadecimal

Condition ou :

-retourne la valeur de la première variable « true »

ARRAY AND STRING FUNCTIONS

onLoad problem :

if you try to select an id for example before the page is fully loaded, you will end up in the best case with an null error, that's because the page isn't yet fully loaded and you're trying to operate on an inexistant id to solve this problem you should make sure that your script will be executed after the page is loaded and that's can be done either by using :

pure javascript : window.load=function(){//script here} (or document.load… read the difference between the both here)

or you can use **jquery** :

$(function(){//script here} which is equivalent to $(document).ready({})

Sources :

http://www.w3schools.com/js/