Internet Software Technologies
JavaScript – part one

IMCNE

A.A. 2008/09

Gabriele Cecchetti

Why introduce JavaScript

- To add dynamicity and interactivity to HTML pages

What's a script

- It's a "little" interpreted program
- It's platform independent

What's JavaScript

- It is object oriented too.
- It is a scripting language for a browser
- It is simpler than Java
- It is loosely typed
- It is prototype based (inherited properties may vary)

JavaScript's goals

- To provide **dynamicity, effectiveness and interactivity** to HTML pages
- To **develop application** on client side;
- To **check data input** from user before submitting them to a server;
- To **manage searches** on client side;
- To store and load **cookies**.

How to include JavaScript

- **Inside HTML document**

```
<script type="text/JavaScript">
<!--
   JavaScript code goes here…
//-->
</script>
```

- **External source file**

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

JavaScript Language version

- You can specify different version scripts to support different browser:

```
<script type=”text/JavaScript” language=“JavaScript1.2”>
```

- Or you can reditect JavaScript not enabled browser to another page:

```
<NOSCRIPT>
<META HTTP-EQUIV REFRESH CONTENT="0;
URL=anotherpage.html">
</NOSCRIPT>
```


Hello world example

```
<html>
<head>
<title> Hello World</title>
</head>
<body>
<script type=”text/JavaScript” >
<--
document.write(“Hello World!”)
// -->
</script>
</body>
</html>
```

When a script will be run ?

- It's dependent by:
  - code position (inside the document)
  - code organization
- Generally it runs from:
  - sequentially, inside <SCRIPT> element;
  - loaded from external file;
  - when an event handler is invoked;
  - when JavaScript code is activated from HTML link.

Where to put the JavaScript ?

- **Scripts in the head section:** Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.
- **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 the content of the page.
- You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

Example

```
<html>
<head>
<title> onLoad event </title>
<script language="JavaScript" >
window.alert(" Not in function "); // this is executed
  first
function loading() {
window.alert("onLoad event");
}
</script>
</head>
<body onLoad="loading()"> <!- call function loading() -
  ->
</body>
</html>
```


CONCEPTS & SYNTAX

JavaScript Statements

- A JavaScript statement is a command to the browser. The purpose of the command is to tell the browser what to do.
- JavaScript is a sequence of statements, which are executed in the sequence they are written.
- JavaScript Statements can be grouped together in blocks {
  }
- JavaScript is case sensitive!

Comments

```
// comment: one line
/* comment…
…
…
(also several lines
…
*/
```

Quotes

- Single ' and double " quotes can be used: note the following example:

```
alert('Don't do this!')
```

- Error →

```
missing ) after argument list.
alert('Don't do this!')
```

- Correct with:

```
alert('Don\' t do this!')
```

- Or

```
alert("Don't do this!")
```


Semicolon

- Semicolon ; is used as a instructions separator.

→ New instructions can start each newline or after a semicolon.

Literals

- Literals are explicit quantities:
  - **Numbers:**
    - **Decimal / Octal (0) / Hexadecimal (0x),**
    - **Floating point and fixed point;**
  - **Boolean** (True/false);
  - String arrays;
  - Null value;
  - Special characters.

newline	\n	carriage return	\
horiz. tab	\t	backspace	\b
linefeed	\f	backslash	\\
single quote	\'	double quotes	\"

Basic Types

- numeric → `12, 113.256`
- string → `"Ciao"`
- boolean → *`true`* or *`false`*
- *`null`* → null value
- *`undefined`* → value non specified
- `function` → function defined or built-in

Variables

- Global variables
  - Valid for the whole HTML document:
  - They must declared>
    - inside `<SCRIPT>` element, preferably in the head section,
    - before they can be used, and
    - not inside a function.
- Local variables
  - Valid only inside the function where they are declared.
- They *can* be prefixed by `var` keyword.
- They can be initialized or leave to null value.
- They do not have any type.

Variables Example

```
<html><head><title> Scoping Rules</title>
<script language="JavaScript">
<!--
var cc=0 ;          // cc: global var.
var dd=scr();       // dd: global var.
document.writeln("Global variale: " + cc);
document.writeln("Local variable: " + dd);
function scr() {
  var cc =3;        // cc now is a local variable
  return cc;
}
// -->
</script></head><body ></body></html>
```

Undefined variable

- If you declare a variable without assigning a value they are undefined; e.g.

```
var mycolor
alert(mycolor);
```

return undefined.

- That is different to assign a *null* value

```
var mycolor=null
```

allocates the memory for that variable assigning the null value.

Variable names

- Only letters, number and underscore can be used as variable names
- The first char must always be a letter or underscore.
- Javascript is case-senditive
- *Do not use JavaScript keywords*.

JavaScript Keywords

```
abstract do if package throw boolean
double implements private throws break
else import protected transient byte
extends in public true case false
instanceof return try catch final int
short var char finally interface
static void class float long super
while const* for native switch with
continue function new synchronized
default goto null this
```


Loosed type variables

- JavaScript variables have not a specified type, so you can declare

  ```
  myvar="hello"   //   or
  myvar=59
  ```

- It's the embedded JavaScript compiler which detect the type.
- Different types of variables can be concatenated:

  ```
  another_var="my dog is " + 7
  ```

- The compiler will make the necessary conversions.

Common ways to convert variables types:

- From number to string:
  - Using quotes (string concatenation);
  - `mystring=toString(number);`
- From string to number:
  - `parseInt("10")`
  - `parseFloat("10.5")`
  - `eval` - The eval() function evaluates a string and executes it as if it was script code; e.g.

    ```
    eval("x=10;y=20;document.write(x*y)");
    ```


Operators

- Arithmetic operators
- Assignment operators
- Concatenation operators
- Comparison operators
- Logical operators
- Conditional operators

Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (division remainder)
++	Increment
--	Decrement

Assignment Operators

=	
+=	x = x + y
-=	x = x - y
*=	x = x * y
/=	x = x / y
%=	x = x % y

Concatenation operator

- Example
  document.writeln(“abc” + “def”);
  produce the same effect of:
  document.writeln(“abcdef”);
- or when you mix strings and numbers:
  document.writeln(“x = ” + 5);
  produce the same effect of:
  document.writeln(“x = 5”);

Comparison operator

Operator	Description	Example
==	Is equal to	X==8 is false
===	Is exacly equal	x===5 is true, while x===“5” is false
!=	Is not equal	x!=8 is true
>	Is greater than	x >8 is false
<	Is less than	x >8 is true
>=	Is greater or equal than	x>=8 is false
<=	Is less than or equal than	x<=8 is true

Example: `if (age<18) document.write("Too young");`

Logical operators

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	Not	!(x==y) is true

Conditional operator

- **Syntax**

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

- **Example**

```
greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

Other operators

- operand1**,** operand2
- **typeof**: **typeof** *operand*;
- **void**: **void** *expression*; or
  void (*expression*)

Flow control statements

- If … Else
- Switch
- For Loop and While Loop
- Break Loops

If and If … else statements

- ```
  if (condition) {
      code to be executed
      if condition is true
  }
  ```
- ```
  if (condition) {
      code to be executed
      if condition is true
  } else {
      code to be executed
      if condition is not true
  }
  ```


Switch

- ```
  switch(n) {
  case 1:
      execute code block 1
      break;
  case 2:
      execute code block 2
      break;
  default:
      code to be executed
      if n is different from case 1 and 2
  }
  ```
  ```

For Loop and While Loop

- ```
  for (var=start; var<=end; var=var+incr) {
      code to be executed
  }
  ```
- ```
  var=start
  while (var<=endvalue) {
      code to be executed
      var=var+incr
  }
  ```


Break and continue

- The break command will break the loop and continue executing the code that follows after the loop (if any). Example:

```
for (i=0;i<=10;i++) {
    if (i==3) { break; }
    document.write("The number is " + i + "<br />");
}
```

- The continue command will break the current loop and continue with the next value. Example:

```
for (i=0;i<=10;i++) {
    if (i==3) { continue; }
    document.write("The number is " + i + "<br />");
}
```

Functions

- A function contains code that will be executed by an event or by a call to that function.
- You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).
- Functions can be defined both in the <head> and in the `<body>` section of a document. However, to assure that the function is read/loaded by the browser before it is called, it could be wise to put it in the `<head>` section.

How to define a Function

- ```
  function functionname(var1,var2,...,varX) {
      some code
      return somevalue // optional
  }
  ```

- A function with no parameters must include the parentheses `()` after the function name.
  ```

Function Example (1/2)

```
<html><head><title> Variable and functions </title>
<script language=”JavaScript” >
<!--
function test( s ) {
   var a=10;
   var c= 10 * s;
   document.writeln(c);
}
// -->
</script></head>
<body onLoad=“test(10)">
</body>
</html>
```


Another Function Example (2/2)

```
<html><head><title> Function Call</title><script
   language="JavaScript">
<!--
document.writeln(concat("1","2","3","4"));
document.writeln(concat("1","2"));
document.writeln(concat("Oggi ","e\'"," Lunedi\'"));
function concat() {
   var fin=" ";
   if (arguments.length <3)
       return “Few Argoments!<br>";
   for(var t=0; t<arguments.length; t++)
       fin += arguments[t];
   fin+="<br>";
   return fin;
}
// -->
</script></head><body ></body></html>
```

Built-in Function

- **eval**(*string)* → eval then execute it
- **isFinite**(*number*) → true, NaN
- **isNaN**(*value*) → true, false
- **parseFloat**(*string*) → float, NaN
- **parseInt**(*string* [, *radix*]) → integer, NaN
- **Number**(*obj_ref*) → number
- **String**(*obj_ref*) → string
- **escape**(*string*) → string to RFC1738
- **unescape**(*string)* → *RFC1738 to string*