University of Oum El Bouaghi

Advanced Web Programming

JavaScript Reminders

Concerned students

Faculty	Department	Level	Speciality
ESNL	MI	B3	ISSE

Interest

- **Example 1**

```
<html>
<head>
   <title>Static Page</title>
</head>
<body>
<div>
   We are on the 11/01/2024
</div>
</body>
</html>
```

Interest

- **Example 1**

```
<html>
<head>
   <title>Static Page</title>
</head>
<body>
<script type = "text/javascript">
 date = new Date();
 document.writeln(" We are on the", date);
</script>
</body>
</html>
```

Some advice

- **JavaScript code can be challenging to debug.**
  - Use a **debugger**,
  - Pay attention to formatting (**code indention**),
  - Never use global variables (use the '**var**' keyword).

Insertion of JS code

- **In the header of the page**
  - Between the **<head>** and **</head>** tags.
  - Code executed **upon a user event**.
  - The event is located **in the body** of the document.

```
<html>
<head>
<script type="text/javascript">
   functionf () { alert('Goodbye'); }
</script>
</head>
<body onUnload="f();"> // closing the current page
</body>
</html>
```

Insertion of JS code

- **In the body of the HTML page**
  - Between the **<body>** and **</ body>** tags.
  - Executed when **loading the page**.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
   alert('good morning');
</script>
</body>
</html>
```

Insertion of JS code

- **In the header or in the file**
  - File in text format.
  - **Adventage:** Reusability of the script across multiple pages.
  - **Disadvantage:** Additional request to the server

```
<html>
<head>
<script type="text/javascript"  src="fichier.js"></script>
</head>
<body>
<input type="button" onclick="popup()" value="Clic">
</body>
</html>
```

JavaScript console

- **JavaScript console** is present in the development tools of Firefox or Chrome(keyboard shortcut **F12**).

Examlpe:

- x; //Undefined variable
- **var** x; x; // Defined variable, uninitialized
- **var** x = 1; x; // Defined and initialized Variable

- // Display a variable in the **console**

```
console.log(x);
```

- // Display an alert dialog

```
alert("Coucou !");
```

Typing and Variables

- Variables are objects.
- They can be declared at anytime.
- '**var**' keyword can be used for their declaration.
- Every value has a type in JavaScript.
- This type is **not specified during the declaration** of a variable.

```
var x = 1;
typeof x;  // → number
var y = "How was your day?";
typeof y;  // → string
```

- **Types are dynamic**

```
var x;                  // x is undefined
var x = 5;                    // x is a Number
var x = "Ahmed";        // Now x is a String
```

Typing and Variables

- In **JavaScript,** There are six data types including:
  - Five primitives: **Boolean, Number, String, Null, Undefined.**
  - Objects: **Object.**

- **Undefined** is the type of variable that have no value (e.g. uninitilized variable).

```
var x;
typeof x;
// → undefined
```

Operators

- **affectation**

  +=, -=, *=, /=, %=, &=, |=, <<=, >>=

- **comparison**

  ==, !=, <, <=, >, >= ,=== ,!==

- **arithmetic**

  %, ++, --

- **logic**

  &&, ||, !

- **bit**

  &, |, ^ (XOR), <<, >>, >>>

Control structures

- Javascript has control structure similar to those in C language:
  - **if else**
  - **for, while, do.. while**
  - **break, continue**
  - **switch**
- **for … in** (foreach) loop for arrays.
- and the **throw, try, catch** exceptions.

Functions

- Declaration like in Java, C++

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

- Variables **can store** functions!

```
var square = function (x) {  return x * x;};
```

- Functions are '**first-class**' **objects**: they can be manipulated and exchanged like any other JavaScript objects.

```
function square(x) {  return x * x; };
var varfonc = square;  // Variable assignment
//Function execution with the operator()
varfonc(2); // → 4
```

Functions

- A function returns, always, something. By default the function returns undefined.

```
function badsquare(x) {
var y = x * x;
        // The developper forgot to write return y;
}
badsquare(2);
        // → undefined
```

- A function can take a function as argument

```
function boum() {alert('Boum!');}
setTimeout(boum,2000);
// setTimeout executes the function in the variable 'boum'
after 2 seconds.
```

Main Functions

- **eval(string)** : Evaluates the Javascript code.
- **Number(var)** : Convert to a number.
- **String(var)** : Convert to a string.
- **int parseInt(string[,radix])** : Convert to an integer based on the specified radix(base).
- **float parseFloat(string)** : Convert to a real.
- **encodeURI(uri)**
- **decodeURI(uri)**

JavaScript timers

- **timers** allow executing an action after a certain delay, or performing an action every x seconds.
- Activating a timer is a simple operation:
- **setTimeout() and clearTimeout()**

```
window.setTimeout("myfunction()",1000);
```

- This code will call the function myfunction() after one second (time is expressed in milliseconds).
- One can stop a timer before its timeout by using **clearTimeout**() function. It is necessary to "name" timeout, like this:

```
mytimer=window.setTimeout("myfunction()",5000);
```

- This will stop it like this:

```
window.clearTimeout(mytimer);
```

JavaScript timers

- **setInterval() and clearInterval()**
- The logic is the same, exept that here we are calling a function at regular intervals.

```
window.setInterval("mafonction()",1000);
```

- This code will call myfunction() function every second until the page is closed or the timer is stopped by the **clearInterval()** function.
- It is necessary to name the timer.

```
mytimer=window. setInterval("myfunction()",5000);
window.clearInterval(mytimer);
```

Datatypes

- **JavaScript has eight Datatypes:**
  - Number, String, Boolean, Bigint, Undefined, Null, Symbol, Object.
- **Example:**

```
var x = 12;
var str = "hello";
var k;
var bool = true;
var y = 12.14;
var tab=[1,2,3];
var p=null;
function f() {}

console.log(typeof x); // number
console.log(typeof str);//string
console.log(typeof k); //undefined
console.log(typeof bool);//boolean
console.log(typeof y);// number
console.log(typeof tab);//object
console.log(typeof p);// object
console.log(typeof f);//function
```

Object creation

Several ways are possible to create objects in Javascript

Choose the method that best suits the needs based on the object's complexity and srtucture.

- **Object Literal:**

```
var student ={
   name:"Ahmed",
   age:20,
   sexe:"m"
}
```

Object creation

- **Using new keyword**

```
var student = new Object();
student.name= "souad";
student.level="2nd year";
student.age="22";

function Student (name,level,age, moyS1, moyS2){
    this.name=name;
    this.level=level;
    this.age=age;
    this.moyS1 = moyS1;
    this.moyS2 = moyS2;
    this.genmoy = function () {return(this.moyS1+this.moyS2)/2;}
}
var student1 = new Student("Adel", "2", "22", 14, 15);
console.log(student1.genmoy());
```

Object creation

- **Using a constructor function**

```
function Person(firstName, lastName, dateOfBirth) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.dateOfBirth = new Date(dateOfBirth);
    this.fullName = function() {
        return this.firstName + " " + this.lastName;
    };
}

var person = new Person("John", "Doe", "04/05/2000");
```

Note: Choose the method that best suits your needs based on the structure and complexity of the object you want to create.

Creating instance using **JSON** format:

- Definition (JSON –JavaScript Object Notation)
- JSON: data format that allows the serializetion of objects.
- Simplified implementation (compared to XML).
- Natively recognized by JavaScript.

JavaScript Objects

- **Using JSON format:**

```
var joe = {
  first_name: "Pascal",
  last_name: "Dulo",
  brothers: [
    { name: "William", age: 36 },
    { name: "John", age: 34 }
  ],
  displayInfo: function() {
    console.log(this.first_name + ' ' + this.last_name);
    console.log("Brothers:");

    for (var i = 0; i < this.brothers.length; i++) {
      var brother = this.brothers[i];
      console.log("Brother" + (i + 1) + ":" + " Age:", brother.age );
    }
  },
};

joe.displayInfo();
```

JavaScript Objects

- Objects **as associative Arrays**.

- In JavaScript, an object is an associative array, with attributes and methods identified by their names.

```
var person = new Object();
person['name'] = "John";
person['age'] = 30;
person['sayHello'] = function() {
    console.log("Hello, my name is " + this.name + " and I am " +
this.age + " years old.");
};
person.sayHello(); // Output: Hello, my name is John and I am 30 years old.
```

Javascript Objects

- **Access to Object Properties**

```
person1 = new Person("Jack", "Dulo", 18);
var firstName = person1.firstName; // dot notation
var lastName = person1["lastName"]; // associative array notation

// Accessing all properties:
for (var i in person1) {
    alert("Attribute: " + i + ", value: " + person1[i]);
}
```

Javascript Objects

Basic Objects

Object	Description
Array	Enables the manipulation of arrays
String	Allows to manipulate strings.
Math	Offers methods for handling common mathematical functions (log, exp, etc).
Date	Allows manipualtion of current date and provides methods for performing operations on date, hours, minutes and seconds.
Boolean	Manipulation of logic values.
Number	Manipulation of numeric values and constants.
Function	Code designed to perform a task.

- **Pseudo Object-Oriented Programming: POOP**

  - Refers to the **use of *OOP techniques in JavaScript***.
  - Pseudo-classes can be created with properties and methods because in JS, there is no concept of inheritance and polymorphism.
  - There are predefined pseudo-classes that can be:
    - **Usual pseudo-classes**: String, Date, Math, Array.
    - **Window relates pseudo-classes:** window, document, history, location.

- **Usual pseudo-classes: "Date"**
  - **Allows panipulation of dates; it only has methods (no properties).**

Method	Description
getFullYear()	Get **year** as a four digit number (yyyy)
getMonth()	Get **month** as a number (0-11)
getDate()	Get **day** as a number (1-31)
getDay()	Get **weekday** as a number (0-6)
getHours()	Get **hour** (0-23)
getMinutes()	Get **minute** (0-59)
getSeconds()	Get **second** (0-59)
getMilliseconds()	Get **millisecond** (0-999)
getTime()	Get **time** (milliseconds since January 1, 1970)

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

- **Usual pseudo-classes: "Array"**
  - Allows the definition of arrays with a single index.

```
matrix = new Array(20); // defines an array with 20 elements numbered from 0 to 19
matrix[15] = 4.56; // value of an element in the array
```

  - This pseudo-class has only one property: **length**, and three methods:
  - **join():** concatenation of all elements into a string (separator to be specified, otherwise a comma).
  - **sort():** sorting with an optional criterion to be specified.
  - **reverse():** transposition.

- **Window pseudo-classes:** Related to the display objects manipulated by the browser. The key classes include:

  - The pseudo class **window**: Display area.
  - The pseudo class **document**: Content of the window.
  - The pseudo class **history**: Stores the sequence of sites visited by their URLs.
  - The pseudo class **location:** Getting the current URL of the page and redirect the browser.

Event Driven Programming

Event	Description
onBlur	Triggered when a select, text, or textarea form item loses focus after user interaction.
onChange	Triggered when the user alters the target's content.
onClick	Triggered when the user clicks on an object.
onFocus	Triggered when a select, text, or textarea item is selected.
onSelect	Triggered when some text in a text box or text area is selected.
onSubmit	Form submission.
onLoad	Triggered when a page or a resource is fully loaded.
onMouseOver	Triggered when the mouse pointer moves over an element.
onMouseOut	Triggered when the mouse pointer moves out of an element.

Event Driven Programming

- **Establishing an event handler:**
  - Use a tag and add the event keyword with javaScript code that specifies the action to be executed if the event occurs.

```
<TAG onSomething="javascript code">
```

References

https://www.cs.uct.ac.za/mit_notes/web_programming/pdfs/chp13.pdf

https://www.heelpbook.net/2014/javascript-events-onblur-onchange-onclick-onfocus-onselect-onsubmit/

https://www.w3schools.com/js/js_syntax.asp