JavaScript Programming

Marie Beurton-Aimar, Florent Grelard, Fabien Baldacci

Master Bioinformatics - 2020-2021

JavaScript

- Context :
  - Script language for Web.
  - Code is execute on the client (browser) side.
  - ***Not related to Java language***.
  - Fundamental characterics, semantir, inherited from SmallTalk, Lisp ou Scheme : first class functions, closure, lambda functions, array more than lists, objects...
  - Functional programming language AND object oriented based on prototypes, interpreted and with dynamic typing.
  - Read the article :

`http://microclub.ch/2012/10/21/javascript-le-langage-de-programmation-le-plus-incompris-du-monde/`

The Tale

- Written in 10 days by Brendan Eich (Netscape company), old name *Mocha/LiveScript*.
- ECMA has produced a specification in 1996-1997.
- ECMAScript is the specification name and JavaScript the name of the most well-known implementation (Version ES6 from 2015).
- 2005 Mozilla and Eich joined ECMA - implementation of ActionScript.
- A second revival for JavaScript with library developments : jQuery, Doja, Mootools ....Become a key tool for dynamic web pages.
- More recently, the framework Nodejs proposes an architecture with JavaScript on the server side and HTML5 on the client side, which allows to open communications, i.e.“sockets”, to collect geographical date for example,

JavaScript Today

- All configurations and materials have a JavaScript “runtime” .
- Is in the core of Firefox Mobile ES et WebOS.
- Most of client applications iOs, Android et Blackberry are based on JavaScript (Flash is planned to disappear).
- On the server side, excellent performances concerning response time.

JavaScript Today - Resume

Main characteristcs

- Functionnal langage : function is the first module to execute.
- Functions are *first class* objects, i.e. functions are considered as objects.
- Consequences, they can be :
  - created using litterals,
  - assigned to object variables or properties.
  - given as parameter,
  - returned as result,
  - properties of objects or methods.
- Event prgramming, *listeners* specifications.
- Events are stored into a *FIFO* list, the browser process these events, in an asynchronus way, calling the associated *listeners*. We call that *callbacks*.

Example !

- Into an `html` file write into the `<head>` block :

```
<script type=''text/javascript''>
        console.log(``Hello World'');
</script>
```

- Into the `<body>` block, place one instruction, for example create a button and associate a `function` to it to display `Hello`

```
<body>
  <b> Example </b>
  <button onclick="alert('Hello');">
      Click on me
  </button>
</body>
```

JavaScript - the language

- All instructions end with `;`
- Blocks are defined by `{ }`
- Comments : `//` or `/* a comment */`
- Declarations :
  - `var myVar;` // global or local variable.
  - `let myVar;` // only local variable (ES6)
  - `const PI=3.14;` //constante (ES6).
  - `myVar=10;` //declaration without the keyword `var` or `let` for global variables ***not recommended***.

Data Types

- Inter, real, exponent..
- Booleans : `true`,`false`
- Character chain : string.
- Array : `array` i.e. `list` en Python.
- Object.

Convention for variable names

- Ascii characters only.
- `Case sensitive`.
- Can not begin with a digit.
- The chars `-+/*@` are forbidden.

Operators

Mathematics

- `+, -, /, *, %`
- `+=, -=, /=, *=, ++, -`

Logic

- `>, < , >=, <=, ===, ===`
- `&&, ||, !`

Alert on the equality test

- `==` cause an automatic conversion :
  `'3' == 3` respond `true`
  `'3' === 3` respond `false`

Method for String

- Redefinition of the operator `+`.
- `length` to obtain the size.
- Liste of classical method :
  - `charAt, charCodeAt, concate, includes, match, indexOf ...`
  - `slice, split, substr, toLowerCase,`
  - `replace, repeat, search ...`

The type Array

- Close to Python list.
- Manage element positions.
- Property : `length`

```
var arr = [''first element'', 'another', 3.14, ''Python''];
var item = arr[0];
arr[5]=''I add an element'';
```

Methods

```
var size = arr.length;
var end = arr[arr.length \1];
var Nsize = arr.push(''toto'');
var last = arr.pop();
```

The type Object

- Creating objects with properties.

```
var obj = {};
var item = {name: ''my name'', age:20};
var aName = item['name'];
var another = item.name;
var anage = item.age;
item.surname =''julie '';
```

Control Structures

- Test `if, switch ... case,`
- Loop `while, do ... while, for`

```
if (condition) {
   instructions;
   ....
}
else if (condition){
   do something;
}
else {
   instructions;
}
```

```
switch (variable){
   case val1: instructions; break;
   case val2: instructions; break;
    default: instructions;
 }
```

Control Structures

```
var i=0;
 while (i<0) {
   instructions;
   i++;
 }

 for (var i=0; i<10;i++) {
      instructions;
 }
```

```
var i=0;
do {
   instructions;
   i++;
} while (i<10);
```

Control Structures - JS5

```
var tableau =[2,3,7,5];
for (var  elem in tableau) {
  console.log(i);
}


var item = {name:'curie', surname:'marie', price:'nobel'};
for (var prop in item) {
    console.log(prop);
    console.log(item[prop]);
}
```

Control Structures - ES6

```
var tableau =[2,3,7,5];
for (let i of tableau) {
  console.log(i);
}

var word='crazybiocomputing';
for (let i of word) {
    console.log(i);
}
```

- Alert : impossible on objects, they are not iterable

Functions

```
function foo(arg1,arg2) {
     var result = arg1 +arg2;
     return result;
}
console.log(foo(2.3);
```

- Parameters are given following the current rules : by value, or by references depending on their own type.
- **Anonymous Functions**: the variable contains an *object function*

```
var bar = function(arg){ result = arg+3;
                         return result;
                       };
var result = bar(4);
```

Optional Arguments

```
function foo1 (arg){
  if (arg == undefined){
     arg=0;
   }
  return arg;
}
foo1(); \\ return 0
foo2(); \\ return 0
foo3();\\ return 0
```

```
function foo2 (arg)  {
     arg = arg || 0;
     return arg;
}
function foo3 (arg=0){
     return arg;
}
```

Programming by event

Javascript allows :

- to do actions on HTML tags,
- to manage the HTML tree structure, i.e. ***DOM tree*** (Document Object Model)

$\Rightarrow$ ***Dynamical*** Web pages - the structure is modified by events.
$\Rightarrow$ Modification of the DOM structure as a response to the events.

`window` and `document`

Two objects are defined by default in Javascript in the context of Web page :

- ***`window`*** : is the browser window in which the document is loaded.
- ***`document`*** : is the DOM loaded in the window (HTML tree)

Example of methods :

- `window.alert()` : open a dialog box (popup) into the Web page
- `document.write()` : write text into the HTML document.

Elements selection

In order to manage HTML elements into Javascript code, first, it is necessary to get them as objects.
HTML elements selection can be done by using :

- ***tags***
- ***`id`***
- ***`class`***
- ***CSS selector***.

Indentity selection

The method ***getElementById('id')*** of an object `document` selects the unic element with the id given as parameter.

Example : change the image in JS:

html file

```
...
<img src="image.jpg" id="
    monImage" />
...
```

JavaScript file

```
var imageJS = document.
    getElementById('monImage');
imageJS.src="newImage.jpg"
```

Several elements selection

Several methods :

- From the ***tag*** name : `getElementsByTagName()`
- From the ***classe*** : `getElementsByClassName()`
- From the ***sélecteur CSS*** : `querySelectorAll()`.

$\Rightarrow$ return an ***array*** of elements

Example : change the image in JS:

Html file

```
...
<h1 class="toleft">
<div class="toleft">
...
```

JavaScript file

```
var elements = document.
    getElementsByClassName('toleft')
    ;
console.log(elements[0]);
```

Manage elements

- Collected objects by Javascript have ***attributes*** : same attributes than in HTML.
- These attributes are reachable both for reading and writting.

Example :

```
var imageJS = document.getElementById('myImage');
imageJS.alt = "Text of description";
console.log(imageJS.alt);
imageJS.src = "newImage.png";
imageJS.className = imageJS.className + "the class";
```

Modification of tag content

- The attribute ***innerHTML*** is the HTML content of an element :
  - reading : contains the tags
  - writting : its content is interpreted (tags are considered as HTML)
- The attribute ***textContent*** is the textual content of an element :
  - reading : does not contained the tags
  - writting : its content is not interpreted (tags are considered as text)

Differences between `innerHTML` and `textContent`

```
<div id="example">
<p> This is  <span> my content </span>
</p>
</div>
```

`innerHTML` :

```
var element = document.
    getElementById("example"
    );
var htmlText = element.
    innerHTML;
console.log(htmlText); //the
    text contains the tags <
    p> and <span>
element.innerHTML = "<p>␣
    lalilou␣</p>";
//the tags are interpreted.
```

`textContent` :

```
var element = document.
    getElementById("example"
    );
var htmlText = element.
    textContent;
console.log(htmlText); // /
    the text does not
    contain the tags<p> and
    <span>
element.textContent = "<p
    >␣lalilou␣</p>"; //
tags are considered as text.
```

Manipulation of the DOM

Possibility to create elemens with***`document.createElement()`***

To modify the DOM tree:

- Adding : `appendChild()`, `insertBefore()`
- Suppressing : `removeChild()`
- Remplacing : `replaceChild()`

Event capture

Goal : Link a function to an event occurrence concerning one element.

The link between an event and an element is done by an ***event listener***.

Example :

Element (HTML) $\rightarrow$	**Evenement (JS)** $\rightarrow$	**Event listener (JS)** $\rightarrow$
Button	Clic	Send data
Image	Clic	Change the image

Events

It exists different event types:

- actions from the keyboard or the mouse: ***click***, ***keyup***, ***mouseover*** . . .
- change the state : ***change***, ***focus***. . .
- end or beginnning of element loading in the page: ***load***.

Event listener (1/2)

- “Listening” events on element.
- Method ***`addEventListener()`*** register a function link to an event for an element.

Methode usage :

```
object.addEventListener(typeEvent, fonctionDeclenchee);
```

- `object` : object target (ex: `document` or object collected by `getElementById()`).
- `typeEvent` : character chain to cite the event.
- `functionOn` : called function (ex: image changing, or tag content)

Event listener (2/2)

Another way to register a function `addEventListener()`:

```
object.onevent = functionOn;
```

Examples :

```
document.onkeyup = functionDeclenchee;
//Equivalent to
document.addEventListener("keyup", functionOn);
```

Event capture : example (1/2)

HTML :

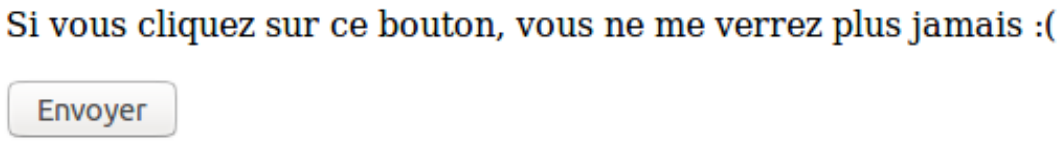
```
...
<p id="myParagraph"> If you click on this button,
    you will never
see me again :(
</p>

<button id="myButton" type="submit"> Send </button>
...
```

Javascript :

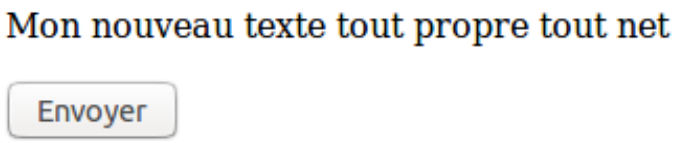
```
var changeMyParagraph = function() {
 var texte = document.getElementById("myParagraph");
 texte.textContent = "My new text very clean ";
}

var button = document.getElementById("myButton");
button.addEventListener("click", changerMyParagraph);
//Equivalent to : button.onclick = changeMyParagraph;
```

Event capture : example (2/2)

At the beginning :



After the click on the button :



Registering several functions

For the same element we can have :

- several registration for different events
- several registration for the same event

Methodology and best pratices :

1. Define a function ***`setupListeners`*** in charge to set the registration
   $\Rightarrow$ make easy the code maintenance.
2. Call the function `setupListeners` when the HTML page is ***totally loaded***.
   $\Rightarrow$ `window.addEventListener("load", setupListeners)`

Object `event`

- An object ***`event`*** is created for each event
- The object type ***`event`*** varies depending on the event
- This object has attributes which provides information about the event type:
  - `type` : type of event (clic, . . .)
  - `clientX`, `clientY`, `screenX`, `screenY`, `pageX`, `pageY` : coordinates relating to the visible part of the page - the screen - or the whole page.
  - `key` : information about the key press
  - `target` : event target.

Object `event` : example (1/2)

```
var changerCouleur = function(event) {
 var texte = document.getElementById("monParagraphe");
 if (event.key === "r") {
  texte.style.color = "red";
 }
 else if (event.key === "g") {
  texte.style.color = "green";
 }
 else if (event.key === "b") {
  texte.style.color = "blue";
 }
}

var setupListeners = function() {
 document.addEventListener("keyup", changeColor);
}

window.addEventListener("load", setupListeners);
```

Object `event` : example (2/2)

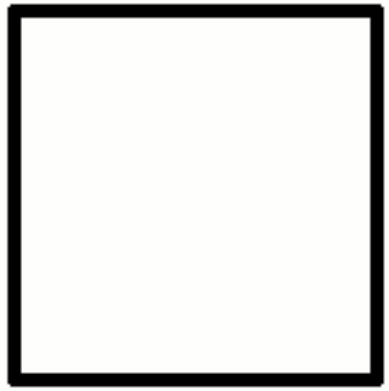
Result :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.

Object `event` : example (2/2)

Result :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Object `event` : example (2/2)

Result :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.

Object `event` : example (2/2)

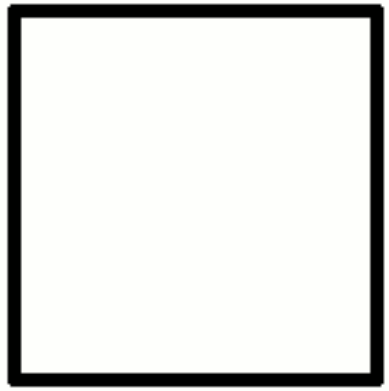
Result :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.

Object `event` : example (2/2)

Result :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Object `event` : example (2/2)

Result :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.

Object `event` : example (2/2)

Result :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.

Object `event` : example (2/2)

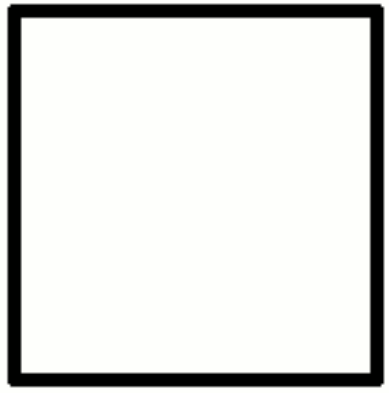
Result :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.

Object `event` : example (2/2)

Result :

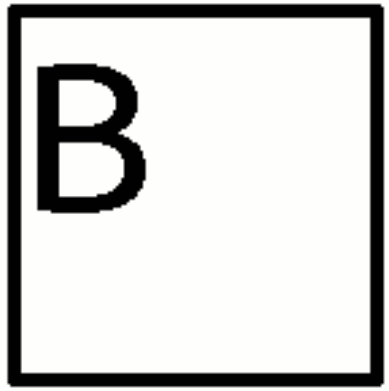
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Object `event` : example (2/2)

Result :

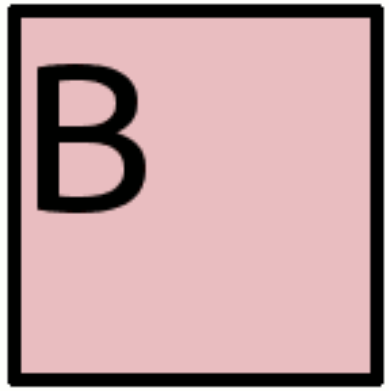
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Object `event` : example (2/2)

Result :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Object `event` : example (2/2)

Result :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.