JavaScript

- Contexte :
  - Langage de script du Web.
  - Execution du code sur la machine/le navigateur du client.
  - ***N'est pas un sous-ensemble de Java***.
  - Caractéristiques intrinsèques, sémantique, héritées de SmallTalk, Lisp ou Scheme : fonctions de première classe, fermeture, fonction lambda, tableaux au lieu de listes, objets... au lieu de listes de propiétés.
  - Langage de programmation fonctionnel et orienté objet basé sur les prototypes, interprété et dynamique.
  - Lire l'article :

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

La petite histoire

- Ecrit en 10 jours par Brendan Eich chez Netscape, avec comme ancien nom *Mocha/LiveScript*.
- ECMA produit une description de la spécification en 1996-1997. ECMAScript est le nom de la spécification et JavaScript la plus connue des implémentations (Version ES5 actuellement pour tous, ES6 en déploiement).
- 2005 Mozilla et Eich rejoignent ECMA - implémentation de ActionScript.
- Un nouveau départ pour JavaScript avec le développement des bibliothèques jQuery, Doja, Mootools ....Devient un outil incontournable pour les pages dynamiques.
- Plus récemment, la plateforme Nodejs propose une architecture avec JavaScript sur le serveur et HTML5 sur le client, permet d'ouvrir des communications de type "sockets", de capturer des données géographiques ....

JavaScript Today

- Tous les types de plateformes et périphériques possèdent en général un “runtime” JavaScript.
- Est au coeur de Firefox Mobile ES et WebOS.
- La plupart des applis coté client, iOs, Android et Blackberry repose sur JavaScript (disparition progressive de Flash).
- Coté serveur, excellentes performances sur les temps de réponse.

JavaScript Today

Caractéristiques principales

- Langage fonctionnel : la fonction est le premier module d'exécution.
- Les fonctions sont des objets de *premier ordre - first class functions*, i.e. les fonctions sont traitées comme des objets.
- Conséquences, elles peuvent être :
  - créées via des littérales,
  - assignées à des variables ou propriétés d'un objet,
  - passées en paramètre,
  - retournées comme résultat,
  - propriétaire d'objets ou de méthodes.
- Programmation événementielle. Spécification de *listeners*, actions.
- Les événements sont dans une *FIFO*, le navigateur traite ces événements, de façon asynchrone, en appelant les *listeners* associés. On parlera de *callbacks*.

Un exemple !

- Dans un fichier `html` placer dans l'élément `<head>` :

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

- Dans le `<body>` placer une instruction, par exemple créer un bouton et associer une `fonction` qui affiche `Hello`

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

JavaScript - le language

- Toutes les instructions se terminent par `;`
- Les blocs sont définis par `{ }`
- Les commentaires : `//` ou `/* un commentaire */`
- Les déclarations :
  - `var myVar;` // variable globale ou locale.
  - `let myVar;` // variable locale uniquement (ES6)
  - `const PI=3.14;` //constante (ES6).
  - `myVar=10;`//declaration sans `var` ou `let` pour des variables globales ***Mauvaise pratique***.

Type de données

- Entier, réel, exposant..
- Booléens : `true`,`false`
- Chaine de caractĕres : string.
- Tableau : `array` i.e. `list` en Python.
- Objet.

Conventions sur les noms de variables

- Caractĕres Ascii seulement.
- `Case sensitive`.
- Ne peut commencer par un chifffre.
- Les caractères `-+/*@` sont interdits.

Opérateurs

Mathématiques

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

Logiques

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

Attention au test d'équalité

- `==` provoque une conversion automatique :
  `'3' == 3` répond `true`
  `'3' == 3` répond `false`

Méthode du type String

- Redéfinition de l'opérateur `+`.
- `length` permet de connaitre la taille.
- Liste de méthodes classiques :
  - `charAt, charCodeAt, concate, includes, match, indexOf ...`
  - `slice, split, substr, toLowerCase,`
  - `replace, repeat, search ...`

Le type Tableau - Array

- Proche des listes Python
- Manipule la position des éléments.
- Propriété : `length`

```
var arr = [``premier element'', 'un autre', 3.14, ``Python''];
var item = arr[0];
arr[5]=''j'ajoute un element'';
```

Méthodes

```
var taille = arr.length;
var fin = arr[arr.length \1];
var Ntaille = arr.push(``toto'');
var dernier = arr.pop();
```

Le type Objet

- Création d'objets ayant des propriétés.

```
var obj = {};
var item = {nom: ''mon nom'', age:20};
var unNmo = item[nom];
var autre = item.nom;
var unage = item.age;
item.prenom =''julie '';
```

Structures de Controle

- Test `if, switch ... case,`
- Boucle `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;
 }
```

Structures de Controle

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

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

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

Structures de Controle - JS5

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


var item = {nom:'curie', prenom:'marie', prix:'nobel'};
for (var prop in item) {
    console.log(prop);
    console.log(item[prop]);
}
```

Structures de Controle - ES6

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

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

- Attention : impossible sur les objets car ils ne sont pas iterable

Fonctions

```
function foo(arg1,arg2) {
     var result = arg1 +arg2;
     return result;
}
console.log(foo(2,3);
```

- Passage des paramètres dans les règles habituelles par valeur ou par référence suivant leur type.

Fonctions Anonymes

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

Arguments Optionels

```
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;
}
```