COURS 1

Les basiques du Javascript

BEQUART VALENTIN

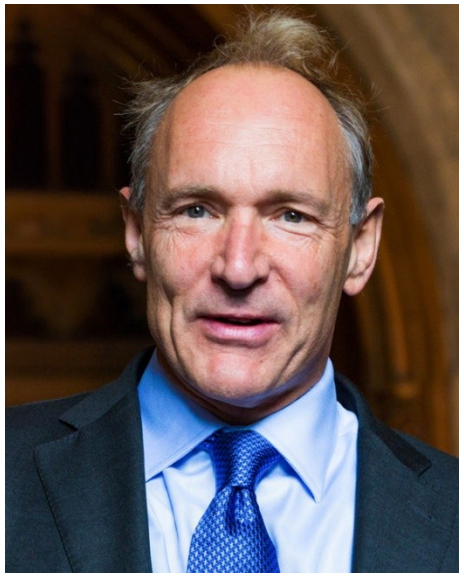
Qui suis-je?

Contact : valentin.bequart@soprasteria.com
Ou valentin.bequart@isen.yncrea.fr
Ou valentin.bequart@externe.junia.com

Un peu d’Histoire…

WORLD WIDE WEB CONSORTIUM (W3C)

Organisme de standardisation -1994
Par Tim BERNERS-LEE



Rasmus LERDORF - 1994

1996 – Hakon WIUM LIE

JAVASCRIPT

Langage à Prototype - 1995
Brendan EICH

Le Javascript, kécéssé??

- Implémentation de l'ECMAScript
  - Ensemble de Norme sur les langages de programmation de types script.

- 1997-1998-1999 : ECMAScript 1, 2 et 3
- 2007 : ECMAScript 4
- 2009 : ECMAScript 5
- 2015 : ECMAScript 6
- 2016 : ECMAScript 7 (Array.protype.include, croyez moi, vous l'adorerez)
- 2017 : ECMAScript 8
- 2018 : ECMAScript 9
- 2019 : ECMAScript 10
- 2020 : ECMAScript 11 (Ajoute des simplifications de vérification d'accès, et l'opérateur ??)
- 2021 : ECMAScript 12 (Ajoute un replaceAll (Attention, Node v15))

A partir d'ici, c'est important

- Un navigateur Web performant et récent :
  - Firefox 55+
  - Google Chrome 61+
  - Chromium 61+
  - Opéra 47+

- Documentation Javascript :
  - https://developer.mozilla.org/

- Editeur de code :
  - Atom, **VsCode** ou autre

JavaScript

Typage
Dynamique
Faible

Orienté Objet
(Prototype)

Interprété

Fonctionnel et
événementiel

TYPAGE DYNAMIQUE FAIBLE

- Faible : Déclarer un type de variable n'est pas requis
- Dynamique : Le contenu d'une variable peut changer de type à volonté

```
let a = 2;
a = "hello world";
var b = 10;
c = false;
```

➢ Une variable se déclare soit sans mots clés, soit avec les mots clés **var**, **let**, **const**…

Déclaration de variables

```
var maVariable1;    // Déclaration d'une variable
var maVariable1 = 42;   // Déclaration + initialisation

// Même chose avec plusieurs variables
var maVariable1, maVariable2;
var maVariable1 = 42, maVariable2 = 'Salut!';
var maVariable1 = 42, maVariable2, maVariable3 = 'Salut!';
```

Instructions

- Comment écrire un programme javascript?

- Série d'instructions
- Délimitées par des points virgules
  (Ils ne sont pas obligatoires, mais c'est une bonne pratique)
- Les blocs d'instructions sont définis par des accolades
  (Parfois, un point virgule se trouve à la fin de l'accolade…)

```
// Une instruction
var a = 3.14;

// Un bloc d'instructions
{
    var b = 'Hello';    // Une instruction
    var c = 42;     // Une instruction
}
```

Conventions d’écritures

- Les variables et les noms de fonction
  - En camelCase
- Un espace entre les blocs logiques
- On indente correctement son code
- On commente son code!
- Des espaces entre les opérateurs
- Les classes en PascalCase

```
/**
 *
 * @description Cette fonction n'a en réalité que peu d'utilité... Comme IE ^-^
 * @param {int} a
 * @param {int} b
 * @returns {int} a
 * @returns {int} c
 */
function myFunc(a, b) {

    let c = a;
    let d = b

    if(a == 1) {
        return 'ok';
    } else {
        for (let i = 0; i < b; i++) {
            c += d;
        }

        return c;
    }

}
```

Tout est objet :

Tout en Javascript est objet, les objets que vous déclarerez, et même les types dit primitifs :

- Les nombres (number)
- Les booléens (boolean)
- Les chaines de caractères (string)
- Les fonctions (function)
- La valeur undefined

```
"Hello world".length;       // 11
(12.3456).toExponential();  // "1.23456e+1"
(function maFonction() {}).name;    // "maFonction"
({toto:42,tata:'Hi!'}).tata;    // "Hi!"
[1,2,3,4].reverse();        // [4,3,2,1]
```

Un Objet?

Un objet :

Un ensemble de paires :
Nom / Valeurs

Un accès en mode « Objet »

Un accès en mode « Tableau »

```
var jedi = {};  // Un objet vide
var jedi = { age:24, name:'Luke Skywalker' };

// Style Objet
jedi.age;       // 24
jedi.name;      // Luke Skywalker


// Style Tableau
jedi['age'];    // 24
jedi['name'];   // Luke Skywalker
```

Un Objet Complexe?

On stocke tout ce que l'on veut :

```
var jedi = {
    age:24,
    name:'Luke Skywalker',
    force: [42,25],
    celebQuote: function(say) {
        return say;
    },
    sabre: {
        couleur: 'bleu',
        taille: 1,
    }
};

jedi.celebQuote("NOOooooooOOn");
jedi.sabre.couleur;
```

Les fonctions

- Le mot clé « function »
- Les accolades
- Les paramètres
- Le return ?
- Appel par le nom

```
function repeat(parameter) {
    console.log(parameter);
    return parameter;
}
```

Les fonctions : Anonymes

- Nommer sa fonction n'est pas obligatoire
  Mais comment peut-on l'appeler?

```
function(parameter) {
    console.log(parameter);
    return parameter;
}
// ??
```

- En la stockant dans une variable :

```
var varRepeat = function(parameter) {
    console.log(parameter);
    return parameter;
};
varRepeat("Echo");
```

Les fonctions : paramètres

Appel d'une fonction classique, avec un paramètre

```
var varRepeat = function(parameter) {
    console.log(parameter);
    return parameter;
};
varRepeat("Echo");
```

Les fonctions : paramètres par défaut (ES6)

```
function repeat(parameter = "Bonjour quand même!") {
    console.log(parameter);
    return parameter;
}

repeat("Echo!"); // Affiche Echo!
repeat(); // Affiche Bonjour quand même!
```

Les fonctions : récupération des paramètres

Classique?

```
function recupParam1(...args) {
    console.log(args);
}
recupParam1(1,2,3,4,5);
// Affiche
//  [1,2,3,4,5]
```

```
function recupParam2(param1, param2, ...rest) {
    console.log(param1);
    console.log(param2);
    console.log(rest);
}
recupParam2(1,2,3,4,5);
// Affiche
//  1
//  2
//  [3,4,5]
```

Avec les « … »?

Un peu plus spicy : Arrow functions

```
function repeat(parameter) {
    console.log(parameter);
    return parameter;
}
```

```
var arrowRepeat = (parameter) => {
    console.log(parameter);
    return parameter;
}
```

Arrow Function

```
var arrowRepeat2 = parameter => {
    console.log(parameter);
    return parameter;
}
```

Un seul paramètre

```
var arrowRepeat3 = parameter => console.log(parameter);
```

Une seule instruction

```
var arrowRepeat4 = parameter => parameter + 'a';
```

Une seule instruction: un return

La portée

var : Déclare une variable locale à une fonction

let (ES6) : Déclare une variable locale à un bloc

```
var value1 = 1;
let value2 = 2;

function maFonction() {

    var value3 = 3;

    if(true) {
        let value4 = 4;
    }

    console.log(value3); // 3
    console.log(value4); // ReferenceError: value4 is not defined

    return;

}
maFonction();

console.log(value1);    // 1
console.log(value2);    // 2
```

Les opérateurs

- <
- >
- =
- ==
- ===
- ||
- &&
- !
- ??

- Inférieur
- Supérieur
- Affectation
- Comparaison
- Comparaison Stricte
- OU
- ET
- NOT
- NULLISH

Blocs conditionnels : IF ELSE

```
if (condition) {
    // Instructions
} else {
    // Autres Instructions
}
```

```
var a = 15;
if ( a < 10 ) {
    console.log("Passe pas");
} else if ( a > 10 ) {
    console.log("Passe");
} else {
    console.log("Sniper");
}
```

Les opérations ternaires

```
let note = 20;
note == 20 ? console.log('Champion(ne)!!') : console.log('Try again...');
```

Permet de s’éviter un bloc if

Possibilité de chainer les ternaires

Blocs conditionnels : SWITCH

```
var a = 1;
switch(a) {
    case 1:
        console.log('1');
        break;
    case 2:
        console.log('2');
        break;
    case 3:
        console.log('3');
        break;
    default:
        console.log('Zéro?');
}
```

Boucles : While et Do While

```
var i = 0;
while(i < 10) {
    ++i;
    console.log(i);
}

var i = 0;
do {
    ++i;
    console.log(i);
} while(i < 10);
```

Connaissez vous la différence?

Le Do While s'exécute toujours au moins une fois!

Boucles : For

```
for(let i = 0; i < 10; ++i) {
    console.log(i);
}
```

Boucles : For – avec la portée

```
var i;
for(i = 0; i < 10; ++i) {
    // Du code...
}

for(var i = 0; i < 10; ++i) {
    // Du code...
}
```

```
for(let i = 0; i < 10; ++i) {
    // Du code...
}
```

Boucles : For In

Itère sur les attributs énumérables d'un objet

Permet de récupérer les attributs

```
for(let key in jedi) {
    console.log(key, jedi[key]);
}
// Affiche :
//age 24
//name Luke Skywalker
```

Boucles : For Of (ES 6)

Permet d'itérer sur des objets dit « itérables » et d'en récupérer les valeurs

```
var myArray = [24,'Luke Skywalker'];
for(let value of myArray) {
    console.log(value);
}

// Affiche :
// 24
// Luke Skywalker
```

L'objet Jedi n'est pas itérable!

```
var jedi = {
    age: 24,
    name:'Luke Skywalker'
};

for(let value of jedi) {
    console.log(value);
}

// TypeError: jedi is not iterable
```

Types Primitifs VS Object

```
let test = new Number(1);
let test2 = new String('test');
let test3 = 1;
let test4 = 'test';
```

Opérateurs : typeof et instanceof

Typeof vous donne le type primitif d'un objet

```
let test = new Number(1);
let test2 = new String('test');
let test3 = 1;
let test4 = 'test';

console.log(typeof test); // object
console.log(typeof test2); // object
console.log(typeof test3); // number
console.log(typeof test4); // string

console.log(test instanceof Number);    // true
console.log(test2 instanceof Number);   // false
console.log(test2 instanceof String);   // true
console.log(test3 instanceof Number);   // false
console.log(test4 instanceof String);   // false
```

Instanceof vous indique s'il est une instance d'un objet donné

Les tableaux

```
var monTableau = [];
var monTableau = [1,2,3];
console.log(typeof monTableau);          // object
console.log(monTableau instanceof Object);  // true
console.log(monTableau instanceof Array);   // true
```

Les tableaux - accès

BIEN

```
var monTableau = ['value1', 'value2', 'value3'];
console.log(monTableau[0]); // "value1"
console.log(monTableau[2]); // "value3"
console.log(monTableau.length); // 3
```

PAS BIEN

```
console.log(monTableau.2);  // "v
```

L’objet Array - opérandes

- Push() : ajoute un élément à la fin du tableau
- Pop() : retire le dernier élément du tableau
- Unshift() : ajoute un élément au début du tableau
- Shift() : supprime le premier élément du tableau

```
var monTableau = ['value1', 'value2', 'value3'];
monTableau.push('value4');  // ['value1', 'value2', 'value3', 'value4']
monTableau.pop();           // ['value1', 'value2', 'value3']
monTableau.shift();         // ['value2', 'value3']
monTableau.unshift('value4');   // ['value4', 'value2', 'value3']
```

L’objet Array – ForEach()

Parcours le tableau

Exécute pour chaque valeur la fonction passée en paramètre

```
var monTableau = ['value1', 'value2', 'value3'];
var maFonction = function(value) { console.log(value); };
monTableau.forEach(maFonction);
// Affiche
//   "v1"
//   "v2"
//   "v3"

var monTableau = ['value1', 'value2', 'value3'];
monTableau.forEach(value => console.log(value));
```

L’objet Array – Map()

Parcours le tableau, exécute pour chaque valeur la fonction passée en paramètre

Retourne un tableau contenant le résultat de la fonction

```
var numbers = [0,1,2,3,4,5,6,7,8,9];
var squares = numbers.map(value => value*value);
console.log(squares);
```

L’objet Array – Filter()

Filtre les éléments du tableau pour lesquels la fonction passée en paramètre retourne true

```
var numbers = [0,1,2,3,4,5,6,7,8,9];
var pairs = numbers.filter(value => 0 == value%2);
console.log(pairs);
```

L’objet Array – Some()

Retourne true si au moins un élément du tableau valide la fonction passée en paramètre

```
[0,2,4,6,8].some(value => 0 == value%2);
[0,1,2,3,4].some(value => 0 == value%2);
[1,3,5,7,9].some(value => 0 == value%2);
```

L’objet Array – every()

Retourne true si tout les éléments du tableau valident la fonction passée en paramètre

```
[0,2,4,6,8].every(value => 0 == value%2);   // true
[0,1,2,3,4].every(value => 0 == value%2);   // false
[1,3,5,7,9].every(value => 0 == value%2);   // false
```

L’objet Date

```
let now = new Date();

let date = new Date("2021-10-07");

let otherDate = new Date(Date.UTC(2021, 9, 7, 8, 30, 0, 0));

let otherDateWithoutUTC = new Date(2021, 9, 7, 8, 30, 0, 0);

console.log(now); // 2021-10-02T14:33:40.140Z
console.log(date); // 2021-10-07T00:00:00.000Z
console.log(otherDate); // 2021-10-07T08:30:00.000Z
console.log(otherDateWithoutUTC); // 2021-10-07T06:30:00.000Z -> Mauvais fuseau Horaire
console.log(otherDate.getFullYear());
console.log(otherDate.getMonth()); // 0 to 11
console.log(otherDate.getDate()); // 1 to 31
console.log(otherDate.getHours());
console.log(otherDate.getMinutes());
console.log(otherDate.getSeconds());
console.log(otherDate.getMilliseconds());
console.log(otherDate.getDay()); // Sunday - 0 to 6
```

Prêtez attention à l’UTC!

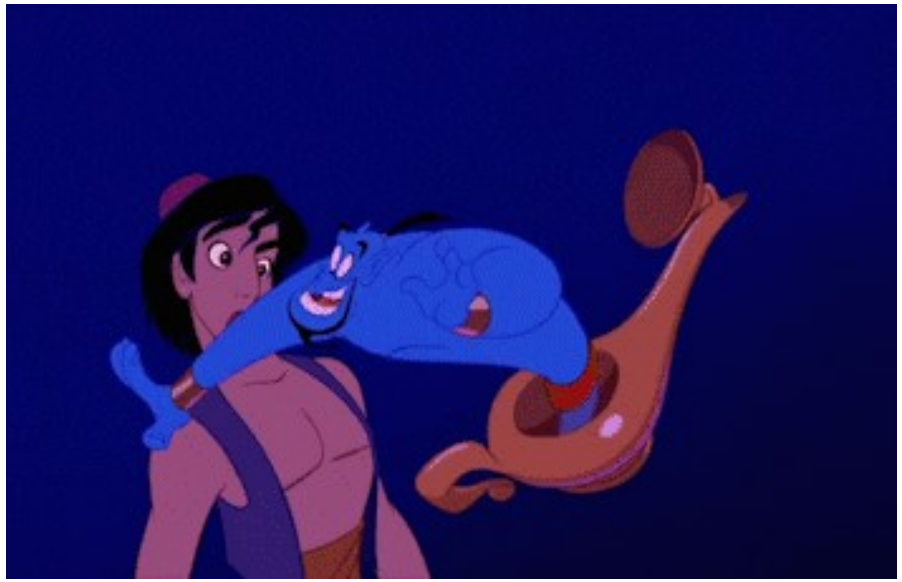
La conversion

```
let str = "20";
console.log(typeof str); // string

let num = Number(str); // Devient un nombre, 20
console.log(typeof num); // number

let test = true;
console.log(typeof test); // boolean

let value = String(test); // value est un String "true"
console.log(typeof value); // string

let ask = Number("j'apporte un café à mon prof de JS");

console.log(ask); // NaN, conversion échouée...

console.log( Boolean(1) ); // true
console.log( Boolean(0) ); // false
```

Les objets natifs les plus utiles :

- Date
- Error
- JSON
- Map
- Math
- Promise
- Proxy
- RegExp
- Set
- String
- WeakMap
- WeakSet

Votre meilleur ami : console.log();



Vous permettra de vérifier la valeur de vos variables au cours de votre algorithme -> Pour vous aider à débugger.

Inclusion dans une page Web

```
<html>
  <head>
    <!-- Les scripts sont chargés avant le body -->
    <script src="./myScript1.js"></script>
    <script src="./myScript2.js"></script>
    <script src="./myScript3.js"></script>
  </head>
  <body>
    <!-- Mon code HTML -->

    <!-- Les scripts sont chargés après le body -->
    <script src="./myScript4.js"></script>
    <script src="./myScript5.js"></script>
  </body>
</html>
```

TIME TO PRACTICE!

Installation de Node

- https://nodejs.org/fr/
- Prendre la version LTS
- Test de version :
- Exécution

```
PS C:\Users\valbe\Desktop\Cours JS\Cours 1\Test> node --version
v10.16.3
PS C:\Users\valbe\Desktop\Cours JS\Cours 1\Test>
```

```
node .\cours1.js
```

Exercice 1 : Pour commencer

- Réaliser les fonctions mathématiques suivantes :
  - Addition
  - Soustraction
  - Multiplication
  - Division
  - Mise au carré

Exercice 2 : Un peu d’ES6

- Reprendre les fonctions précédentes :
  - Addition
  - Soustraction
  - Multiplication
  - Division
  - Mise au carré

  - … Et en faire des Arrows Functions!

Exercice 3 : Un peu d'Array

- A l'aide du tableau fourni :

```
let myArray = [157, 10, 81, 1000, 4, 1024, 16, 492, 9, 287, 0];
```

- Réalisez les trois exercices suivants :
  - Extraire tous les nombres impairs du tableau
  - Générer un nouveau tableau qui contient le sinus des valeurs de myArray
  - Déterminer la quantité de nombres dont le logarithme en base 10 correspond à leurs indices dans le tableau.

Exercice 4 : Quelques facts..

- A l'aide de la chaîne de caractère fournie :

```
let fact = "Vérité sur Chuck Norris : Hulk s'est battu contre Chuck Norris une fois. Depuis, il fait de la pub pour du maïs.";
```

- Réalisez les quatre exercices suivants :
  - Réaliser une fonction qui remplace le terme « Chuck Norris » par un autre nom
  - Donner la taille moyenne des mots du texte (indice : on sépare au niveau des espaces)
  - Réaliser une fonction qui retourne un tableau contenant chaque caractère unique de la chaîne.
  - Réaliser une seconde fonction qui trie ce tableau dans l'ordre alphabétique.

Exercice 5 : Aidons le BDE

- Le BDE Monarch'ISEN prépare une soirée. Pour l'occasion, ils ont décidé de préparer un nouveau cocktail signature, le Royal'ISEN.
- D'après les sondages réalisés auprès des étudiants, 78% des participants à la prochaine soirée souhaiteront gouter à ce nouveau breuvage.
- On attend environ 250 participants, sur une fourchette aléatoire de plus ou moins 50 personnes.
- Le BDE, pour être certain de la quantité de cocktail à préparer (et ne pas gaspiller son budget) afin de satisfaire cette demande de 78%, demande à la classe de CIR2 de réaliser un algorithme capable de « simuler » plusieurs soirées, afin d'obtenir un nombre moyen de cocktail à préparer pour satisfaire au mieux la demande.

    - Pensez que chaque personne présente à la soirée a donc 78% de chance de choisir le fameux cocktail…

Exercice 6 : Un classique

- Savez vous à quoi sert la fonction Histogramme?
  - Réalisez la fonction Histogramme en JavaScript.
    - Un tableau en entrée, et l'affichage du graphique en console!

Par exemple avec le tableau ci-dessous :

```
let myArray = [20,1,13,8,10,6,15,25,2,10,14,18,9];
```

Merci de votre attention!