



miam

Web

Janvier-avril 2023

JUNIA

HEI · ISEN · ISA

Grande  
école  
d'ingénieurs

# I. Les bases du javascript

# Mots clefs

Frontend

→ html (le squelette)

→ css (la peau)

→ javascript (les muscles)

Backend

Asynchrone

chrome

mozilla

Orienté Objet  
(Prototype)



JavaScript

Interprété

Fonctionnel et  
événementiel

## 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;
    }
}
```

# 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 « ... »?

# 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 chaîner les ternaires

# Boucles : For

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

# 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
```

# 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); // "value3"
```

# 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
```

Votre meilleur ami : `console.log()`;

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

# 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>
```



miam

## Contact

[alexis.tonnoir@miam.tech](mailto:alexis.tonnoir@miam.tech)

06 33 74 64 10

# A propos de Miam

Miam réinvente et simplifie les courses alimentaires.



Qui sommes-nous ? Une startup tech lilloise, qui développe depuis 2019 des solutions d'intelligence artificielle à destination de la grande distribution.

Plus de détails sur <https://miam.tech>