

# JAVASCRIPT

Durée Totale du Module : 21H

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

# JAVASCRIPT

Durée Totale du Module : 21H

## Présentation de Javascript

7

Langage de programmation	8
Multi Environnement	8
Orienté Objet	10
Web Dynamique	10
<b>Algo / Syntaxe JS Moderne (ES6 et +)</b>	<b>13</b>
Amélioration de la DX	13
Setup de Javascript	14
Setup Classique	14
Côté Template	15
Async ou Defer	15
Les Variables	17
Anatomie d'une variable	17
Déclarer une variable et assigner une valeur	17
Plusieurs types de variable	18
Exercice :	18
Solution possible	19
Nombres / Calculs /Modulo ☺/ Incrémenter / Assignement Composé	20
Exercice Calculs - Nombres	20
Solution possible	21
Chaînes de caractères	22
Exercice Phrase	23
Solution possible	24

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Les tableaux	25
Exercice : Arrays	25
Solution possible	26
Exercice array 2 les fonctions pour les tableaux	27
Solution possible	28
Exercice arrays 3	29
Solution possible	30
Fonctions / Fonctions Fléchée / return / param / param par default / scope	31
Function	31
Paramètre	31
Exercice Fonction 1	32
Solution	33
Return	34
Paramètre par défaut	34
Scope	35
Exercice : Quiz 1 Trouver le bug	35
Exercice : Quiz 2 Trouver le bug	36
Exercice : Moyenne	36
Solution	37
Opérateurs de comparaison / condition ternaire	38
Opérateurs de comparaison	38
Condition / opérateur ternaire	39
Condition IF ELSE	40
Exercice : If Else	40
Solution	41
Objets	42
hasOwnProperty	43
Exercice : Objects	43
Solution	44
Boucle While / For / ForEach / For ...of / For ...in / map	45
While	45
FOR	46

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

ForEach	47
For ... of	48
For ... in	49
Convertir des objets en tableaux	50
Exercice : boucles	51
Solution	52
Spread Operator	55
Point sur var let ou const	57
Exercice : Quizz Var	57
Exercice : Quizz let	58
Exercice : Quizz function-var	58
Solution function var	59
Exercice : Quizz if-var	60
Exercice Quizz : if-let	62
Solution if-let	63
Const	63
Gestion des erreurs	64
Try ... Catch	64
Gestion des exceptions avec Throw	67
Finally	70
Les Classes	71
Syntaxe	72
L'héritage	73
Exo Class IMC	75
Exo Class PME	76
Exo Class COMPTES BANCAIRES.	78
La guerre des langages	80
Asynchrone	81
API	81
Fetch()	82
Code Asynchrone	82
Async await	83

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.



Exercice : Contacter une API	85
Then catch	86
Gestion des erreurs Response ET Promise	87
Javascript Modulaire	88
Web Workers	89
<b>D.O.M</b>	<b>90</b>
Présentation	90
Sélectionner des éléments du DOM	91
getElement(s)By	91
QuerySelector()	93
Placer des éléments du DOM	94
insertBefore()	94
append() / appendChild()	95
removeChild()	96
Créer - Modifier - Supprimer des éléments du DOM	97
createTextNode()	97
createElement()	98
Exercice : Créer des éléments - Afficher un profil utilisateur	99
Solution Possible	100
Solution possible	102
Modifier les attributs des éléments du DOM	103
Modifier attribut style	103
Modifier attribut class	104
Fonction classList	105
Modifier ou créer n'importe quel attribut	105
Gérer les évènements du DOM	106
Exercice : réagir au click	107
Solution possible	108
Exercice réagir au click 2	109
Solution possible	110
Exo : réagir au click 3 (capter l'event)	111
Solution Possible	112

<b>Auteur :</b> Jean-François Pech	<b>Date création :</b> 03/03/2023	<b>Relu, validé &amp; visé par :</b> <input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> Mathieu PARIS	<b>Date révision :</b> 10/03/2023	 	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.
---------------------------------------	--------------------------------------	---	--------------------------------------	--	---

Exercice : réagir au focus et au Blur	113
Solution possible	114
Exercice : réagir au Load	115
Exercice : réagir à mouseleave	117
Solution Possible	118
Exercice : réagir au scroll	119
Solution possible	120
Exercice : réagir au clavier	121
Solution Possible	123
Exercice : Réagir au clavier 2	124
Web Storage	125
La fonction setItem()	126
La fonction getItem()	126
Exercice : LocalStorage	127
Solution Possible	129
Exercice DOM Settimeout	130
B.O.M	132
<b>JS &amp; Firebase</b>	<b>133</b>
Set-up de firebase	134
Setup-JS	141
Setup-HTML	143
Setup-CSS	144
Exercice : CREATE : Ajouter un nouvel utilisateur	145
Exercice : READ (Lecture de la BDD : Tous Les Utilisateurs)	146
Exercice : READ (Lecture d'un élément de la BDD : Afficher 1 utilisateur)	147
Exercice : READ (Lecture d'un element de la BDD : Pré-remplir le formulaire d'un utilisateur)	148
Exercice : UPDATE : Sauvegarder les modifications d'un utilisateur	149
Exercice : DELETE (Supprimer un utilisateur)	150
<b>Conclusion</b>	<b>151</b>

## Présentation de Javascript

JS

Voici Brendan Eich (une sorte d'humain)



Nous sommes à l'été 1995, dans les bureaux de la Netscape Communications Corporation, le web est en effervescence grâce aux langages HTML et CSS, en parallèle les applications de bureau sont très populaires pour les différents OS (Windows, MacOs, Linux ...) et le langage de programmation Java propose une machine virtuelle (très populaire) permettant de s'affranchir des barrières liées au langages spécifique des OS.

Brendan va donc travailler sur la problématique de créer un nouveau langage qui devra répondre aux contraintes suivantes :

Un langage de programmation

Multi Environnement

Orienté Objet

Permettant de rendre des pages web dynamiques

10 Jours plus tard naissait Javascript...

(Plus récemment il est à l'initiation du projet de navigateur Brave)

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Langage de programmation

Brendan s'est inspiré de nombreux langages de programmation, notamment de Java mais en simplifiant la syntaxe pour les débutants.

## Multi Environnement

L'aspect multi environnement réside dans le fait que JS s'exécute au sein du navigateur web de l'utilisateur, et tout le monde utilise un navigateur (encore + aujourd'hui) et on retrouve les navigateurs partout (Desktop, Mobile, Tablettes, TV, Montres ...) (\*Il existe des frameworks JS comme Electron permettant de créer des apps desktop à partir d'un app web.)

En effet les navigateurs sont composés de 2 moteurs : un moteur de rendu (Render Engine) ainsi qu'un moteur JS (Javascript Engine).



BLINK

V8

QUANTUM

SPIDERMONKEY

TRIDENT

CHAKRA



BLINK

V8

WEBKIT

NITRO

EDGEHTML

CHAKRA

JS s'exécute dans le navigateur certes mais côté client, Front-End, sur la page HTML qui est déjà chargée.

Par la suite après la version ES6 de JS, on peut exécuter du JS côté serveur (Back-End) via des technologies comme Node.js ou encore Deno.

**Auteur :**

Jean-François Pech

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date création :**

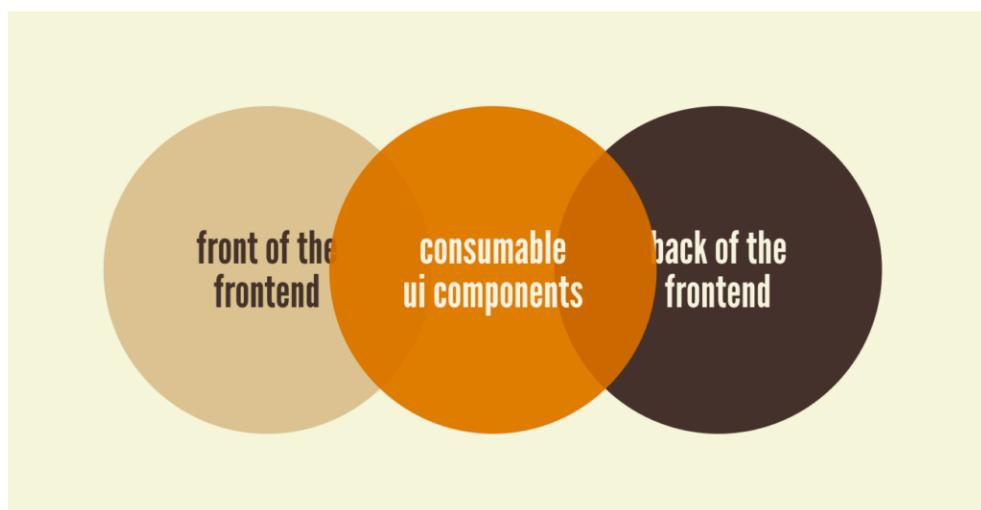
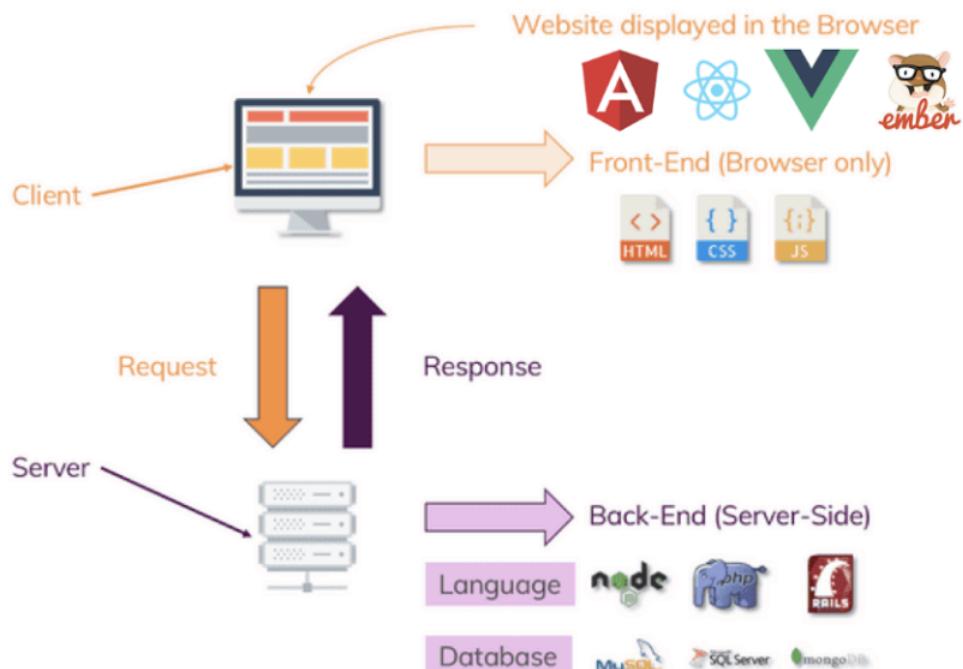
03/03/2023

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.



**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

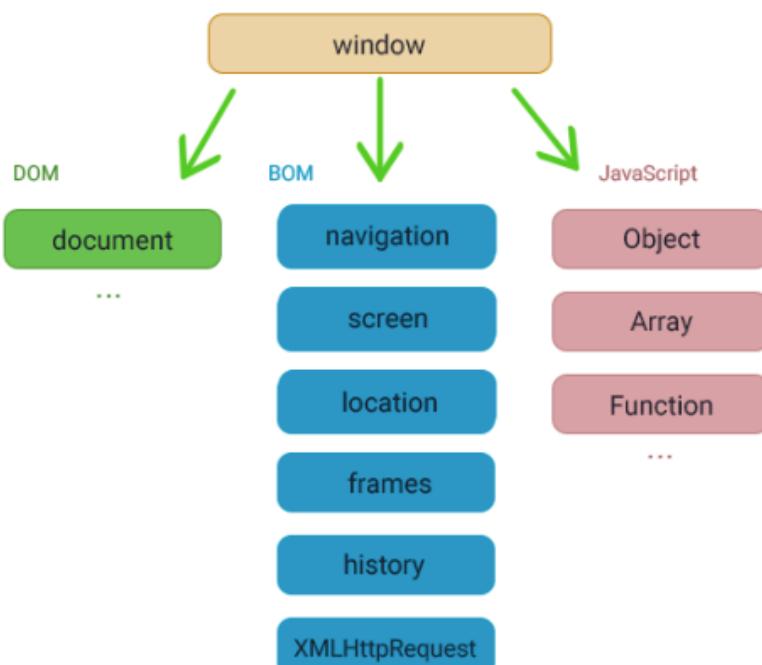
## Orienté Objet

Javascript est un langage conçu pour être orienté objet, à tel point que l'on peut entendre dire « en JS tout est objet », c'est en ce point que JS diffère par rapport à d'autres langages comme Java, pour être plus précis, JS est un langage de programmation orienté objet à prototype, cela signifie que JS va fournir les outils de base du langage (créer une chaîne de caractères, une fonction etc..) via un système d'objet là où Java utilise des classes (en JS une chaîne de caractères, une fonction et même une classe sont des objets).

## Web Dynamique

JS va apporter un ensemble d'outils pour manipuler ce qu'il se passe dans la page web, au sein du navigateur de l'utilisateur.

Techniquement JS contient déjà des objets (avec des propriétés et des fonctions) pour tous les éléments affichés dans la fenêtre de l'utilisateur, l'objet window dans lequel va être exécuté notre code JavaScript (nos variables, nos fonctions, nos classes, nos tableaux, etc.) qui pourra manipuler les objets du BOM (Browser Object Model) pour contrôler l'historique ou la navigation et manipuler les objets du DOM (Document Object Model), les éléments HTML et CSS chargés sur la page.



### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

10/03/2023

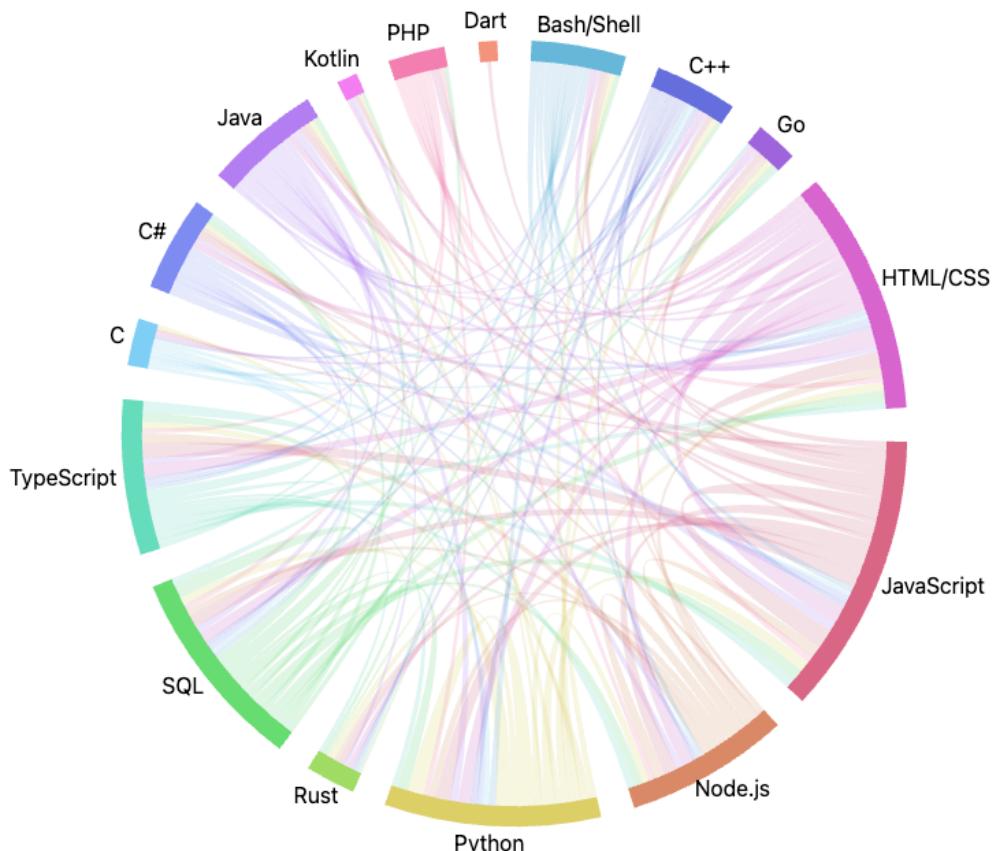


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

### JS : Actuellement

Très vite JS s'est imposé comme un langage incontournable du Web, en combinaison à HTML et CSS dans un premier temps pour améliorer le design des site web, puis au travers de librairies populaires comme JQuery et les évolutions majeures de ES6 qui ont amené de nouvelles technologies côté serveur (Node / Deno) mais aussi les Frameworks (React, Angular, Vue, Svelte...)

<https://stackoverflow.blog/2021/08/02/2021-stack-overflow-developer-survey-results/>



#### Auteur :

Jean-François Pech

#### Date création :

03/03/2023

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date révision :

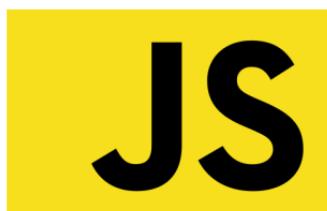
10/03/2023



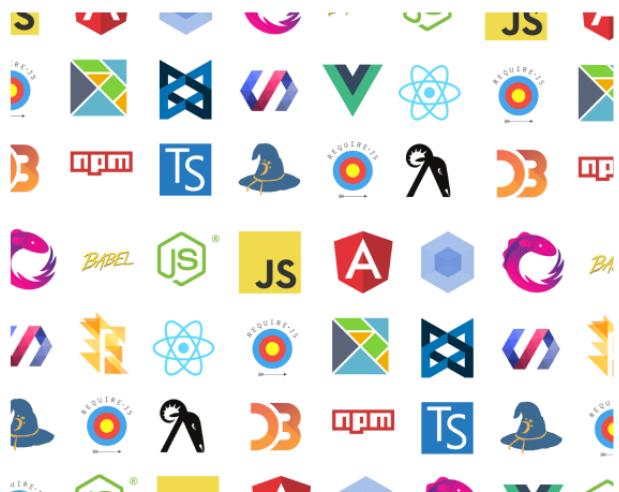
Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

L'évolution fulgurante de JS (un incontournable du web ?)

2008



2021



Un site pour faire l'état de l'art de Javascript :  
<https://2022.stateofjs.com/en-US/libraries/>

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Algo / Syntaxe JS Moderne (ES6 et +)

Dans cette section nous allons nous intéresser aux notions de base de la syntaxe de Javascript.

### Amélioration de la DX



Visual Studio Code : super paramétrable, gratuit, plein d'extensions  
Visual Studio Code .dev : la version web de vscode (c'est cool on peut le connecter direct à GitHub, ya aussi la plupart des extensions) (vs code depuis une tablette ? Smartphone ? À tester)

Alternatives :

Webstorm

Fleet

#### En ligne

codepen  
Stackblitz

### Extensions vscode



LiveShare : Pour partager / streamer son code (travail collaboratif / débug)



GitGraph : Pour mieux visualiser les repositories git sur vscode



Better Comments : Pour des commentaires en couleur  
(ça bug un peu avec des frameworks mais pour JS ça va)



**Vite** : Un ensemble d'outils très utiles pour facilement et ‘vite’ créer puis initialiser des projets (très utilisé avec les principaux frameworks qui peuvent de base comporter beaucoup de fichiers)

Le fonctionnement se fait via des CLI (des lignes de commandes dans le terminal).

Il est nécessaire d'installer node.js sur votre serveur ou votre ordinateur (cela va créer un serveur de développement web)

## Setup de Javascript

Il nous faut un éditeur de code, par exemple [Visual Studio Code](#) et pour plus de confort une extension permettant de simuler un serveur de développement [Live Server](#)  
Et bien entendu nous aurons besoin d'un navigateur internet.(basé sur chromium, Firefox, Safari, etc...)

## Setup Classique

Dans l'approche la plus commune on va bien faire attention de relier le fichier JS à la fin du fichier HTML (juste avant la fermeture de la balise <body>)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Intro JS</title>
</head>
<body>
    <script src="app.js"></script>
</body>
</html>
```

Et un petit script dans le fichier Javascript

```
console.log('Bienvenue dans Javascript');
```

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Côté Template

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Intro JS</title>
</head>
<body>
    <!-- On peut aussi faire du JS directement ici --&gt;
    &lt;script&gt;console.log('Hello World')&lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>

```

## Async ou Defer

On peut aussi placer la balise script pour relier le fichier JS dans le haut du HTML, dans la balise <head> MAIS il faut ajouter l'attribut async ou defer

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Intro JS</title>
    <script src="app.js" async></script>
</head>
<body>
</body>
</html>
```

<https://www.alsacreations.com/astuce/lire/1562-script-attribut-async-defer.html>

**Defer** : Antérieur à la vague HTML5, l'attribut defer existait déjà dans les "anciennes" versions d'Internet Explorer. Il signifie que le navigateur peut charger le(s) script(s) en parallèle, sans stopper le rendu de la page HTML. Contrairement à async, l'ordre d'exécution des scripts est préservé, en fonction de leur apparition dans le code source HTML. Il est par ailleurs reporté à la fin du parsing du DOM (avant l'événement DOMContentLoaded). De nos jours, cet attribut présente moins d'intérêt car les navigateurs disposent par défaut de techniques internes pour télécharger les ressources en parallèle sans nécessairement attendre les autres.

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

**Async** : Nouveau venu dans HTML5, async signifie que le script pourra être exécuté de façon asynchrone, dès qu'il sera disponible (téléchargé). Cela signifie aussi que le navigateur n'attendra pas de suivre un ordre particulier si plusieurs balises <script> sont présentes, et ne bloquera pas le chargement du reste des ressources, notamment la page HTML. L'exécution aura lieu avant l'événement load lancé sur window et ne sera valable que pour les scripts externes au document, c'est-à-dire ceux dont l'attribut src mentionne l'adresse.

Ce comportement est bien pratique pour gagner en temps de chargement, il faut cependant l'utiliser avec prudence : si l'ordre n'est pas respecté, un fichier exécuté de façon asynchrone ne pourra attendre le chargement d'un précédent, par exemple s'il en utilise des fonctions voire un framework. Il ne faudra pas non plus compter appeler document.write() pour écrire dans le document HTML puisqu'il sera impossible de savoir à quel moment les actions seront déclenchées.

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Les Variables

Pour rendre une application web dynamique, il nous faut manipuler, interagir avec des données, par exemple le nombre de billets restant pour un concert, une commande avec un numéro de produit, un prix et une date, une adresse mail, etc. Un langage de programmation va donc utiliser des variables pour enregistrer ces données (pour l'ordinateur c'est un espace mémoire).

### Anatomie d'une variable

Une variable va se définir par 3 aspects :

- Un NOM (pour identifier ce que contient la variable)
- Un TYPE (un nombre, une chaîne de caractère, etc...)
- Une VALEUR (c'est le contenu de la variable)

JavaScript est un langage dont le typage est **faible** et **dynamique**. Cela signifie qu'il n'est pas nécessaire de déclarer le type d'une variable avant de l'utiliser. Le type de la variable sera automatiquement déterminé lorsque le programme sera exécuté. Cela signifie également que la même variable pourra avoir différents types au cours de son existence

[https://developer.mozilla.org/fr/docs/Web/JavaScript/Data\\_structures](https://developer.mozilla.org/fr/docs/Web/JavaScript/Data_structures)

### Déclarer une variable et assigner une valeur

⚠ : Pour assigner une VALEUR à une variable il faut au préalable que cette variable soit initialisée (avec le mot clé let ou const suivi du NOM de la variable).

PS : Prendre aussi l'habitude de finir une instruction JS avec ;

```
let maVariable;
```

Ensuite nous pouvons lui assigner une **VALEUR**.

```
maVariable = 'Hello World';
```

C'est crucial car cela permet à cette variable d'être utilisée dans le programme, auquel cas notre code va provoquer des erreurs et donc entraîner le crash de notre application.

99% du temps nous allons déclarer une variable et lui assigner une valeur directement.

```
let maVariable = 'Hello World';
```

#### Auteur :

Jean-François Pech

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date création :

03/03/2023

#### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Plusieurs types de variable

Le dernier standard ECMAScript définit 8 types de données :

Sept types de données primitifs:

- Booléen (true / false)
- Null
- Undefined
- Number
- BigInt (proposition pour ES2020)
- String (chaine de caractère)
- Symbol (type introduit avec ECMAScript 6)
- Objet

---

### Exercice :

Déclarer, initialiser et afficher en console plusieurs variables de chaque type.  
(chaines de caractères, nombre, nombre à virgule, tableaux, objets)

Bonus : Faire une variable qui contient une fonction dans laquelle on fait un log console  
« Hello World »

---

#### Auteur :

Jean-François Pech

#### Date création :

03/03/2023

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution possible

<https://github.com/jefff404/cours-js/tree/2-variables>

```
/***
 * ****
 * 2-VARIABLES
 * ****
 */
//! On déclare une variable avec let ou const (ou var dans les anciennes versions de JS)
let maVariable;
//! On assigne une valeur à une variable avec le signe =
maVariable = 'Hello World';
console.log(maVariable);

//? Les types de variables (JS utilise un typage dynamique)
let uneString = 'MDR';
let unNombre = 11;
let unBooleen = true;
let unTableau = [1,2,3,'Hello'];
let unObjet = {
    propriete1 : 22,
    propriete2:'LOL'
};
//! Spoiler : on déclare une fonction comme ceci ↪
function testFunction (){
    console.log('Fonction de Test ?');
}

//? Bonus💡 : on peut placer une fonction dans une variable
let uneFonction = function maFonction (){
    console.log('Fonction qui affiche HelloWorld');
}

console.log(uneString);
console.log(unNombre);
console.log(unBooleen);
console.log(unTableau);
console.log(unObjet);
console.log(uneFonction);
```

### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Nombres / Calculs /Modulo ☺/ Incrémenteur / Assignment Composé

Bien entendu, une des utilités principales des langages de programmation c'est de pouvoir faire des calculs (+ ou - complexes) pour cela nous allons utiliser des opérateurs de calculs (+, -, \*, /) comme en Mathématiques.

Une autre technique très largement utilisée dans les applications web consiste à cumuler des chiffres.

Imaginons une variable qui nous sert de compteur de (vues, likes, lectures, etc...) que nous allons incrémenter (c'est-à-dire rajouter une valeur à ce compteur).

Pour cet exemple de compteur dans le cas où l'on veut toujours ajouter 1

```
unCompteur = 0;  
/* unCompteur + 1  
unCompteur ++;
```

On peut également faire avec l'opérateur moins

```
unCompteur --;
```

Si l'on souhaite ajouter de 10 en 10

```
/* unCompteur = unCompteur + 10  
unCompteur +=10;
```

### Exercice Calculs - Nombres

Mettre en place un programme qui affiche en console le résultat de différents calculs (en utilisant tous les opérateurs de base et des nombres à virgule)

En plus faire un console log d'un calcul ultra complexe (utilisant l'ensemble des opérations de calcul).

Mettre en place un compteur et utiliser tous les opérateurs d'assignment composé.

## Solution possible

<https://github.com/jefff404/cours-js/tree/3-calculs>

```
/**  
 * *****  
 * 3-CALCULS  
 * *****  
 */  
console.log(42*675);  
let unChiffre = 9;  
let unNombre = 33;  
console.log(unChiffre*unNombre);  
console.log(2,9+1,3);  
console.log(2.9+1.3);  
console.log((1+1)-(2*3)/2);  
console.log(10%2);  
let unCompteur = 0;  
console.log(unCompteur+1);  
unCompteur = 0;  
unCompteur = unCompteur+1;  
console.log(unCompteur);  
unCompteur = 0;  
/* unCompteur = unCompteur + 1  
unCompteur +=1;  
console.log(unCompteur);  
unCompteur = 0;  
/* unCompteur + 1  
unCompteur ++;  
console.log(unCompteur);  
/* unCompteur - 1  
unCompteur --;  
console.log(unCompteur);  
/* unCompteur = unCompteur + 10  
unCompteur +=10;  
console.log(unCompteur);  
/* unCompteur = unCompteur x 10  
unCompteur *=10;  
console.log(unCompteur);  
/* unCompteur = unCompteur / 10  
unCompteur /=10;  
console.log(unCompteur);  
/* unChiffre puissance 9  
console.log(unChiffre**9);
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

- Jérôme CHRETIENNE
- Sophie POULAKOS
- Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Chaînes de caractères

En javascript nous pouvons tout aussi bien gérer des chaînes de caractères, cela peut constituer juste un mot, une phrase ou tout un paragraphe par exemple pour cela ci-dessous nous allons voir les différentes syntaxes permettant de gérer des strings (chaînes de caractères).

Nous allons donc voir les simple quote, les double quote, (guillemets simple ou double) ainsi que les littéraux de gabarits (ou template strings) qui facilite l'affiche d'une variable au sein d'un texte.

```
let bonjour = 'Bonjour';
```

```
let unMessage = "Bienvenue";
```

```
let welcome = `Bienvenue`;
```

(Alt gr + 7)

Dans certains cas il peut être utile d'assembler plusieurs chaînes de caractères, c'est le principe de la **concaténation**, pour cela on utilise l'opérateur + (à tester 😊)

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice Phrase

Le client, le restaurant "La Pizzeria Raffinata" (**le client insiste sur les guillemets**) nous a choisi pour réaliser son application mobile dans laquelle on peut directement commander en livraison.

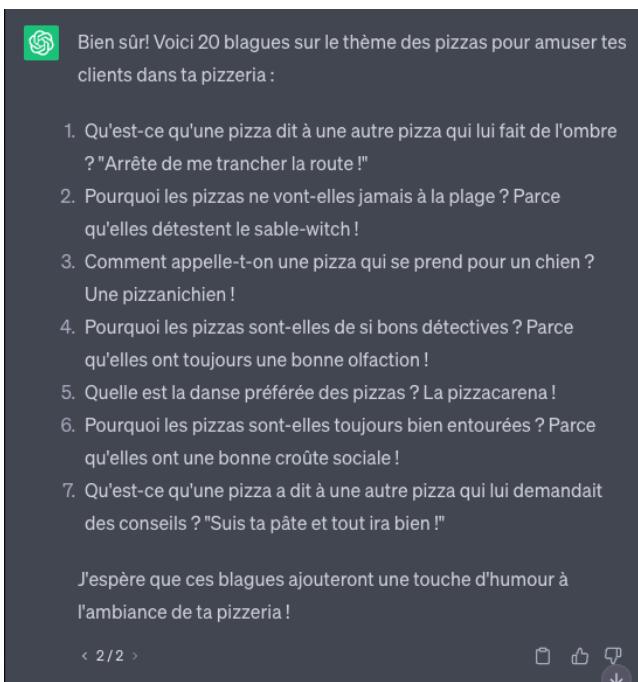
**Définissez selon vous** toutes les variables pertinentes qui résume la commande d'un utilisateur chez "La Pizzeria Raffinata"

Vous devez faciliter le travail pour l'équipe du Template et ranger toutes ces variables dans une variable qui se nommera **SumUpOrderPhrase**, cette phrase devra contenir (on utilise les variables précédentes pour former une phrase) :

Merci d'avoir commandé chez "La Pizzeria Raffinata"

**Ps :**

**Le client nous a envoyé ce message ultérieurement**



Bien sûr! Voici 20 blagues sur le thème des pizzas pour amuser tes clients dans ta pizzeria :

1. Qu'est-ce qu'une pizza dit à une autre pizza qui lui fait de l'ombre ? "Arrête de me trancher la route !"
2. Pourquoi les pizzas ne vont-elles jamais à la plage ? Parce qu'elles détestent le sable-witch !
3. Comment appelle-t-on une pizza qui se prend pour un chien ? Une pizzanichien !
4. Pourquoi les pizzas sont-elles de si bons détectives ? Parce qu'elles ont toujours une bonne olfaction !
5. Quelle est la danse préférée des pizzas ? La pizzacarena !
6. Pourquoi les pizzas sont-elles toujours bien entourées ? Parce qu'elles ont une bonne croûte sociale !
7. Qu'est-ce qu'une pizza a dit à une autre pizza qui lui demandait des conseils ? "Suis ta pâte et tout ira bien !"

J'espère que ces blagues ajouteront une touche d'humour à l'ambiance de ta pizzeria !

< 2 / 2 >

🕒 🙌 📁

« Pour le message de la commande j'aimerais aussi que cela intègre soit le message 1 ou le 7 mais en respectant le saut de ligne.

Par avance Merci. »

## Solution possible

<https://github.com/jefff404/cours-js/tree/4-strings>

```
/**  
 * *****  
 * 4-STRINGS  
 * *****  
 */  
let bonjour = 'Bonjour';  
let unMessage = "Bienvenue";  
let welcome = `Bienvenue`;  
console.log(bonjour,unMessage);  
let unTexte = "Bonjour \"MR Gingle\"";  
let unTxt = 'Aujourd\'hui';  
console.log(unTexte,unTxt);  
let concatenation = bonjour + " et " + unMessage + ', il fait beau temps' + unTxt;  
console.log(concatenation);  
let uneTemplateString = `Hello ! ${bonjour} et ${unMessage}  
on retourne à la "ligne" plus besoin des 'antislash'`;  
console.log(uneTemplateString);
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Les tableaux

Dans des applications qui gèrent plusieurs données il peut s'avérer judicieux de les ranger dans un tableau (dans une seule donnée on aura plusieurs informations).

Pour la syntaxe des tableaux nous assignons à une variable, des crochets [ ], c'est à l'intérieur de ces [ ] que l'on peut ranger des données (des strings, des numbers, des variables ... bref tout type de données)

Ci-dessous un exemple d'un tableau qui contient des nombres :

```
let mesNombres = [100, 200, 300];
```

Les tableaux ont cet aspect pratique qu'ils ont un système d'indexation (on peut accéder à chaque case du tableau) en utilisant cette syntaxe :

```
mesNombres[2]
```

Ci-dessus on accède au contenu de la case numéro 2 du tableau qui contient le nombre...300

(⚠ le système d'indexation des cases d'un tableau commence à 0, donc la case numéro 0 est en fait la première case du tableau).

## Exercice : Arrays

```
///! EXO 4 ARRAYS
//TODO: Créer 1 variable pour un nom,
//TODO: Créer une variable pour un âge,
//TODO: Créer une variable passions qui est un tableau qui contient 2 chaînes de
caractères (au choix)
//TODO: Puis créer une variable tabUser qui est un tableau qui contient les variables du
nom, de l'âge et passions
//TODO: en Console on affiche le tabUser
//TODO : en passant par tabUser on veut afficher en console uniquement les passions
//TODO : en passant par tabUser on veut afficher en console uniquement la 2ème passion
```

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution possible

```
/***
 * ****
 * 5-ARRAYS
 * ****
 */
let prenom = 'JoSé';
let age = 45;
///! On déclare un tableau avec cette syntaxe : []
///! On peut placer ce que l'on veut dans un tableau
let unTableau = [12, 'Salut ca va bien?', prenom, age];
console.log(unTableau);
console.log(unTableau[2]);
///! Exemple avec un tableau dans un tableau
let mesPassions = ["Boxe", "Jonquilles"];
let monPerso = [prenom, age, mesPassions];
///! Avec le système d'index / indice on peut accéder
///! au contenu d'une case du tableau.
///! La 1ère case du tableau est à l'indice 0.
console.log(monPerso[2]);
console.log(monPerso[2][1]);
monPerso[0] = 23;
monPerso[1] = 'YOLO';
monPerso[2][1] = 'COOL';
console.log(monPerso);
///! On peut utiliser la propriété length,
///! .length sur le tableau en lui même
///! nous renvoi le nombre de case du tableau
console.log(monPerso.length);
///! .length sur une case précise du tableau
console.log(monPerso[2].length);
let mesNombres = [100, 200, 300];
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice array 2 les fonctions pour les tableaux

```
//!EXO 4.3 Ajout f° .push()  
//TODO : créer un nouveau tableau qui contient des trucs  
//TODO : allez se renseigner la f° push()  
//TODO : utiliser push pour ajouter un nouveau truc au tableau  
//TODO: On affiche en console ce tableau
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution possible

(la fonction `push()`, peut accepter plusieurs paramètres en les séparant par des virgules pour ajouter plusieurs données en utilisant une seule fois la fonction.

```
let testTabAjout = [10,10,10];
console.log('Avant Ajout :',testTabAjout);
testTabAjout.push('Cortex',92,'Les Pyramides');
console.log('Après Ajout : ',testTabAjout);
```

Pour les tableaux il existe également la fonction `.pop()`, qui va permettre de supprimer le dernier élément du tableau :

```
// pop pour suppr le dernier element du tableau
testTabAjout.pop();
console.log('suppression : ',testTabAjout);
```

```
Après Ajout :                                         app.js:37
▶ (6) [10, 10, 10, 'Cortex', 92, 'Les Pyramides']

suppression :                                         app.js:42
▶ (5) [10, 10, 10, 'Cortex', 92]
```

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice arrays 3

```
///! EXO 4.4 ARRAYS Récap
// TODO: Créer 2 variables leNom et lePrénom
//TODO: Créer un tableau laPhrase et on y ajoute via push, Le nom ,Le prénom Les
initiales
//TODO: Afficher le tableau dans la console le nom le prénom et les initiales
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution possible

```
let leNom = 'Garcia';
let lePrenom = 'José';
let initiales = lePrenom[0] + leNom[0];
let laPhrase = [];
laPhrase.push(leNom, lePrenom, initiales);
console.log(laPhrase);
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Fonctions / Fonctions Fléchée / return / param / param par default / scope

Dans tous les langages de programmation on retrouve le concept de fonctions, il s'agit d'un bloc de code (un sous-programme dans notre programme si l'on veut) qui va contenir plusieurs instructions, de cette manière plutôt que d'écrire plusieurs fois certaines lignes de code, on va les regrouper au sein d'une fonction, de cette manière nous pourrons exécuter ailleurs dans le programme toutes les lignes de code de la fonction

### Function

Syntaxe de base pour déclarer une fonction

```
function maSuperFonction(){
    console.log('Hello World');
    console.log(22+33);
}
```

Ensuite quelque part dans le programme il va falloir exécuter la fonction :

```
///! Détailler la fonction OK, mais ne pas oublier
///! d'exécuter au moins une fois dans le programme cette fonction
maSuperFonction();
```

### Paramètre

Les fonctions ont aussi un concept de paramètre, dans le cas où les instructions au sein de la fonction ont besoin d'une variable extérieure. Ci-dessous une fonction qui va afficher en console une variable unNom

```
///! Certaines fonction ont besoin de prendre un paramètre ici num
///! Pas besoin de déclarer le paramètre, il sera défini à l'utilisation de
///! la fonction
function fonctionAvecParametre(num){
    console.log('Hello World');
    console.log(22+num);
}
///! Ici notre paramètre num aura pour valeur 9
fonctionAvecParametre(9);
```

La variable num est définie directement lors de l'utilisation de la fonction.

#### Auteur :

Jean-François Pech

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date création :

03/03/2023

#### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice Fonction 1

```
///! EXO 5 : Function
// TODO : créer une fonction qui prend un nombre en paramètre
// TODO : La f° doit afficher en console: 33 + le nombre reçu en
paramètre
// TODO : créer une autre fonction qui prend 2 nombres en
paramètre
// TODO : Cette seconde f° doit afficher en console l'ADDITION
des 2 nombres reçus en paramètre
```

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution

```
function fonctionUn(unTruc){  
    console.log(33+unTruc);  
}  
fonctionUn(7);
```

 On peut aussi prévoir des fonctions nécessitant plusieurs paramètres comme ceci

```
function fonctionDeux(unTruc,unBidule){  
    console.log(unBidule+unTruc);  
}  
fonctionDeux(10,88);
```

**Auteur :**

Jean-François Pech

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date création :**

03/03/2023

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Return

Les fonctions gèrent également le concept de return, c'est-à-dire que nous pouvons écrire des fonctions qui vont retourner quelque chose (une variable, un résultat, etc.). Dans les exemples précédents, si on analyse bien, nos fonctions font deux choses techniquement : un calcul ET l'affichage du résultat de ce calcul, mais admettons, nous voulons garder un code clair et précis, on veut une fonction qui fait uniquement du calcul, l'affichage du résultat sera géré ailleurs dans le programme.

Il faut donc adapter notre fonction pour qu'elle retourne un résultat que l'on stockera dans une variable.

Exemple avec une fonction qui a pour but de soustraire 2 nombres :

```
///! Dans certains cas une fonction doit pouvoir retourner quelquechose
///! le résultat d'un calcul par exemple
///! Ci-dessous on fait une fonction de calcul, notre fonction ne fait que ça
///! Elle se charge JUSTE de faire un calcul
///! L'affichage du résultat se fera en dehors de la fonction
function calculReturn(unNombre, unAutreNombre){
    return unNombre - unAutreNombre
}
///! Ici le calcul qui est return par la fonction est stocké dans une variable
///! résultat
let résultat = calculReturn(22,99);
console.log(résultat);
// ou executer la fonction quand on a besoin
console.log('Le résultat : ', calculReturn(22,99));
```

---

## Paramètre par défaut

Une bonne pratique lorsque l'on écrit des fonctions (surtout en travail collaboratif), consiste à renseigner un paramètre par défaut dans le cas où on oublie de renseigner un paramètre quand on exécute une fonction.

```
/** Bonne Pratique : paramètre par défaut
function fonctionAvecParametre(num=0){
    console.log(22+num);
}
```

### Auteur :

Jean-François Pech

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date création :

03/03/2023

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Scope

⚠ Quand on code des fonctions (quand les scripts se complexifient), il est nécessaire de prendre en compte la notion de scope soit la portée des variables.  
Une fonction va pouvoir utiliser des variables déclarées globalement mais le programme global ne peut pas utiliser une variable déclarée dans une fonction.

```
// ? La notion de scope (la portée d'une variable)
// ? Dans l'exemple ci-dessous on a 2 fois la même variable testScope1 qui est déclarée
?????
// ? En fait même si elles ont le même nom ce ne sont pas les même espaces mémoires qui
sont alloués
// ? let testScope1 = 99; est dans le scope global de notre programme
// ? let testScope1 = 12; est dans le scope de la fonction
let testScope1 = 99;
function maFonctionTestScope(){
    let testScope1 = 12;
    console.log('scope de la fonction :',testScope1);
};
maFonctionTestScope();
console.log('scope hors de la fonction :',testScope1);
```

## Exercice : Quiz 1 Trouver le bug

```
//TODO : Pourquoi ca beug ?
function buggyFunction() {
    let wtf = 9;
    console.log(wtf);
};
buggyFunction();
console.log(wtf);
```

✖ ▶ **Uncaught ReferenceError: wtf is not defined** [app.js:71](#)  
**at [app.js:71:13](#)**

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : Quiz 2 Trouver le bug

```
//TODO : Pourquoi ca beug / Pourquoi ca marche pas ?  
let something = 44;  
function functionBugParent() {  
    let something = 9;  
    console.log(something);  
    console.log(lesNews);  
  
    function functionBugEnfant() {  
        let lesNews = `il est 99h67`;  
    }  
};  
functionBugParent();  
console.log(something);
```

```
✖ ► Uncaught ReferenceError: lesNews is not defined      app.js:79  
                                         at functionBugParent (app.js:79:17)  
                                         at app.js:86:1
```

## Exercice : Moyenne

```
//! EXO 5.2 : La moyenne de 2 notes  
//TODO: Créer une fonction qui calcule la moyenne de 2 notes  
//TODO: Afficher le résultat en console
```

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution

```
let noteSport = 8;
let notePhilo = 2;
let laMoyenne = moyenne2notes(notePhilo,noteSport);
// On peut executer la f° AVANT de la définir (pas d'ordre pour décrire les fonctions)
function moyenne2notes(a,b){
    return (a+b)/2;
};
console.log('La moyenne des 2 notes : ',laMoyenne);
```

<https://github.com/jefff404/cours-js/tree/6-functions>

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Opérateurs de comparaison / condition ternaire

### Opérateurs de comparaison

Autre concept qui sera surtout utilisé pour les conditions, ce sont les opérateurs de comparaison cela renvoi un booléen (true ou false).

```
/**  
 * *****  
 * 7- Les opérateurs  
 * *****  
 */  
//! Les booléens : 2 états possibles TRUE ou FALSE (vrai ou faux)  
let a = 11;  
let b = 99;  
console.log("variable a:",a);  
console.log("variable b:",b);  
//! avec == on demande si a est égal à b  
console.log("c'est égal ? :",a == b);  
//! pour vérifier si a est différent de b on utilise !=  
console.log("C'est inégal ? :",a != b);  
//! Ensuite on retrouve les même opérateurs qu'en Mathématique  
//! ici on demande si a est strictement supérieur à b  
console.log("Strictement supérieur ? :",a > b);  
//! ici on demande si a est strictement inférieur à b  
console.log("Strictement inférieur ? :",a < b);  
//! ici on demande si a est inférieur ou égal à b  
console.log("Inférieur ou égal ? :",a <= b);  
//! ici on demande si a est supérieur ou égal à b  
console.log("supérieur ou égal ?:",a >= b);  
//? Attention : de base JS ne prend pas en compte le typage des variables :  
//? ci dessous le nombre 2 est égal au caractère "2" 🤔  
console.log("le chiffre 2 = \"2\"?:",2 == "2");  
//! Pour prendre en compte le type des données que l'on compare, on utilise l'opérateur  
====  
//! c'est l'égalité stricte  
console.log("égalité stricte ?:",2 === "2");  
//! il y a aussi l'inégalité stricte avec l'opérateur !==  
console.log("inégalité stricte ?:",2 !== "2");
```

à noter la différence pour vérifier une égalité entre l'opérateur == et ===, le triple égale va permettre de vérifier aussi si les variables comparées sont du même type.

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Condition / opérateur ternaire

Une syntaxe pour écrire des conditions simples qui renvoient quelque chose ou quelque chose d'autre si une condition est remplie ou non. Grâce à l'opérateur « ? »

Avant le ? on décrit notre condition, après le ? nous avons ce qui est return quand la condition est remplie puis « : » et ce qui est return quand la condition n'est pas remplie

```
//!-----CONDITIONS TERNAIRES-----  
// ? on combine un opérateur de comparaison et l'opérateur ? pour établir une condition  
qui return une chose ou une autre chose  
// ? cela permet de faire une condition if (simple) avec une syntaxe racourcie  
let whatIsYourAge = 6;  
console.log(whatIsYourAge >18 ? '💡': '🧙');  
// Astuce pour check si une variable est définie (si ya QQchose dans son espace  
mémoire)  
let userPremium;  
// On check si une variable est définie la condition c'est juste uneVariable ?  
console.log(userPremium?'OK 👍':'Not OK 🤦');  
// ↑ c'est l'équivalent de ↓  
console.log(userPremium ==true?'OK 👍':'Not OK 🤦');  
// on doit lui assigner QQCHOSE  
userPremium = 'YES';  
console.log(userPremium?'OK 👍':'Not OK 🤦');
```

On peut aussi combiner plusieurs conditions avec les opérateurs  
|| (une condition OU une autre condition), && (une condition ET une autre condition)

```
// ? On peut utiliser des opérateur aussi pour combiner des conditions && (pour ET) ||  
(pour OU)  
console.log(3==3&&3<4);  
let typeUtilisateur = 'Extra';  
console.log(typeUtilisateur == 'Extra' || typeUtilisateur == 'Premium');
```

### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Condition IF ELSE

Avec if, nous allons pouvoir exécuter du code seulement si une condition est remplie, on peut combiner **if** avec **else** qui correspond à SINON

Dans l'exemple ci-dessous nous avons une fonction qui prend un nombre en paramètre et à l'intérieur de cette fonction nous avons plusieurs conditions :

SI le nombre est supérieur ou = à 30 alors on return une phrase

SINON SI le nombre est inférieur à 10 alors on return une autre phrase

SINON on return une troisième phrase

```
//!-----CONDITION avec IF ELSE-----  
// ? Avec if on va pouvoir créer un bloc de code qui s'exécute si une condition est  
remplie  
function calculTableResto(nombreDeReservation) {  
    if (nombreDeReservation>=30){  
        return 'il nous reste pas beaucoup de tables, ça serait pour combien de personnes  
?';  
    }  
    else if(nombreDeReservation<10){  
        return 'Il nous reste une table'  
    }...  
    else{  
        return 'On est fermé !'  
    }  
};  
console.log(calculTableResto(50));
```

---

### Exercice : If Else

```
///! EXO 7 - IF ELSE  
// TODO: Créer une fonction qui reçoit un tableau de 3 notes et qui calcule une moyenne  
entre ces 3 notes  
// TODO: Dans cette f°, SI la moyenne est supérieure ou égale à 15 on renvoi une string  
(très Bien)  
// TODO: Dans cette f°, SINON SI la moyenne est supérieure ou égale à 10 on renvoi une  
string (assez Bien)  
// TODO: Dans cette f°, SINON renvoi une string (refus)
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou  
rediffusion, totale ou partielle, de ce document  
ou de son contenu par quelque procédé que ce  
soit est interdite sans l'autorisation expresse,  
écrite et préalable de l'ADRAR.

## Solution

```
function msgMentionBacOfficiel(tabNotes) {  
    let moyenneCalc = (tabNotes[0]+tabNotes[1]+tabNotes[2])/tabNotes.length;  
    console.log('la Moyenne au Bac : ',moyenneCalc);  
    if (moyenneCalc>=16) {  
        return "Tu as Gagné !"  
    } else if (moyenneCalc >=10 && moyenneCalc<16) {  
        return 'Assez Bien'  
    } else {  
        return 'YO T NUL GRO'  
    }  
};  
console.log(msgMentionBacOfficiel([13,6,3]));
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Objets

Autre type de variable utile pour stocker dans une variable plusieurs informations, les objets avec la syntaxe en accolades {}, assez similaires aux tableaux (mais les objets n'ont pas de système d'indexation), c'est à nous de définir les propriétés (les clés) et leur assigner avec « : » une valeur.

```
// ? syntaxe { unePropriété:uneValeur }
// ? dans un objet on assigne avec : plutot qu'avec =
let user = {
  id:3657826,
  'name':'Seagal',
  firstName:'Steven',
  badges:[ '💻', '🌐', '🎸', '🎧', '🎤' ]
};
console.log(user);
```

On peut accéder aux propriétés d'un objet avec la notation en point

```
console.log(user.name);
console.log(user.id);
```

Ou avec la notation en tableau associatif

```
console.log(user['id']);
```

Pour ajouter une propriété on fait une assignation de valeur (si la propriété existe sa valeur est écrasée par la nouvelle, sinon cela crée la propriété)

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

On peut également effacer une propriété d'un objet avec delete

```
// ? On peut supprimer une propriété
delete user.badges;
```

```
// ? On peut ajouter simplement des propriétés dans un objet avec une assignation de valeur
// ? Si on assigne à une propriété déjà existante cela écrase la valeur
// ? Mais Si on assigne à une propriété non existante cela crée la propriété
user.dps = 9999;
```

## hasOwnProperty

JS propose plusieurs fonctions natives utilisables sur des objets notamment. hasOwnProperty(), qui renvoi true ou false pour vérifier si la propriété d'un objet existe.

Ci-dessous on utilise la fonction dans un console.log() directement.

```
// ? une f° native de JS pour connaître les propriétés d'un objet, hasOwnProperty()
let menuDuJour={
    entree:"Bassine d'Aioli",
    plat:"Rat Adulte",
    dessert:'île Fidji'
};
console.log(menuDuJour);
console.log(menuDuJour.hasOwnProperty('entree'));
```

true

app.js:31

## Exercice : Objects

```
// ! EXO 8 OBJECTS
// TODO : Faire l'exo avec le User et les passions en mode objet
// (un objet user avec des propriétés pour le nom age et passions
qui est lui aussi un objet avec 2 propriétés)
```

### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution

```
// TODO : Faire l'exo avec les passions en mode objet
let nameUser = 'Dong Rodrigue';
let ageUser = 65;
let objetUser = {
    'nom' : nameUser,
    'age' : ageUser,
    'passions': {
        passion1:'Le Drift',
        passion2:'Jonquilles'
    }
};
console.log('objet de utilisateur : ',objetUser);
console.log(objetUser.nom);
console.log(objetUser['passions']); // c le tableau passions
console.log(objetUser.passions.passion2);

objetUser.name = 111;
objetUser.age = 'DonDiegoDelavega';
objetUser.passions.passion2 = 'Le Cinéma';
```

Comme pour les tableaux, dans un objet les propriétés peuvent être réassignées à une valeur

<https://github.com/jefff404/cours-js/tree/9-objets>

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Boucle While / For / ForEach / For ...of / For ...in / map

Comme dans tous les langages de programmation Javascript a un système de boucle, de base cela va nous permettre de répéter des instructions de code selon une condition. Les boucles vont également s'avérer utile par la suite, pour parcourir des itérables comme des tableaux ou des objets.

### While

Correspond à répéter une ou plusieurs instructions TANT QUE une condition est vraie.

```
let unIndex = 0;
while (unIndex < 10) {
    console.log("Le Nombre : " + unIndex);
    unIndex++;
};
```

Ci-dessus on a un index initialisé à 0 et TANT QUE cet index est strictement inférieur à 10 ALORS, on va faire un console.log(), puis ne pas oublier d'incrémenter l'index pour pouvoir passer à une itération de boucle suivante.

```
Le Nombre : 0    app.js:20
Le Nombre : 1    app.js:20
Le Nombre : 2    app.js:20
Le Nombre : 3    app.js:20
Le Nombre : 4    app.js:20
Le Nombre : 5    app.js:20
Le Nombre : 6    app.js:20
Le Nombre : 7    app.js:20
Le Nombre : 8    app.js:20
Le Nombre : 9    app.js:20
```

#### Auteur :

Jean-François Pech

#### Date création :

03/03/2023

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## FOR

Autre manière de créer des boucles, avec `for()`, dans les paramètres on va pouvoir directement initialiser un index, définir une condition et incrémenter l'index dans l'exemple ci-dessous nous allons faire une boucle visant à parcourir chaque case d'un tableau pour l'afficher en console.

```
let listeFilm = ['Ultime Décision', 'Mission Alcatraz', 'Attack Force'];
//? Boucle for, on définit un index (ici c'est i),
//? puis on définit une condition qui va définir le nombre de fois que le code dans la
boucle sera exécutée
//? puis on définit comment on passe à la prochaine itération de la boucle (ici i++, on
augmente de 1 le numéro de la case que l'on console.log)
for(i=0;i<listeFilm.length;i++){
    console.log('boucle FOR : ', listeFilm[i]);
};
```

boucle FOR : Ultime Décision	<a href="#">app.js:14</a>
boucle FOR : Mission Alcatraz	<a href="#">app.js:14</a>
boucle FOR : Attack Force	<a href="#">app.js:14</a>

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## ForEach

Une autre alternative, la fonction `forEach` de JS automatise le parcours d'un tableau ou objet (sans que l'on ait à gérer un système d'indexation (`i++`))  
`forEach` va prendre en paramètre une fonction, cette même fonction pourra avoir un paramètre qui correspondra à chaque case parcourue. (Généralement dans la parenthèse de `forEach` on passe une fonction fléchée).

```
let listeFilm = ['Ultime Décision', 'Mission Alcatraz', 'Attack Force'];
//? La méthode forEach() permet d'exécuter une fonction donnée sur chaque élément du tableau.
// ? On va choisir une variable temporaire pour parcourir les éléments du tableau
listeFilm.forEach(unFilm => console.log('boucle forEach Tableau : ',unFilm));
```

Ici chaque case du tableau sera stockée temporairement dans `unFilm`.

boucle forEach Tableau : Ultime Décision	app.js:27
boucle forEach Tableau : Mission Alcatraz	app.js:27
boucle forEach Tableau : Attack Force	app.js:27

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## For ... of

Encore une alternative pour parcourir des variables tableaux (et autre) c'est la boucle for ... of, qui de la même manière que dans l'exemple précédent dans lequel on va définir une variable temporaire pour parcourir chaque case du tableau :

```
for(let unElementTableo of listeFilm){  
    console.log('boucle FOR...OF : ',unElementTableo);  
};
```

boucle FOR...OF : Ultime Décision	app.js:35
boucle FOR...OF : Mission Alcatraz	app.js:35
boucle FOR...OF : Attack Force	app.js:35

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## For ... in

Si l'on prend le cas des objets JS propose aussi un équivalent à for of (pour les variables de type Array), les boucles for in qui ont exactement la même utilisation que l'exemple précédent

```
const userData = {
    name: 'John Doe',
    email: 'john.doe@example.com',
    age: 25,
    dob: '08/02/1989',
    active: true
};
```

Il faut définir une variable temporaire qui stockera les clés (propriétés) de l'objet

```
// on définit une variable temporaire pour parcourir le objet :
for(let cleObjet in userData){
    console.log(`boucle FOR...IN (objet) : clé:${cleObjet} - valeur : ${userData[cleObjet]}`);
}
```

Ici durant le parcours de l'objet chaque propriété ou clé seront stockées temporairement dans la variable cleObjet.

Rappel : pour accéder aux propriétés d'un objet on la notation en tableau associatif  
unObjet[quelque chose]

```
boucle FOR...IN (objet) : clé:name - valeur : John Doe
boucle FOR...IN (objet) : clé:email - valeur : john.doe@example.com
boucle FOR...IN (objet) : clé:age - valeur : 25
boucle FOR...IN (objet) : clé:dob - valeur : 08/02/1989
boucle FOR...IN (objet) : clé:active - valeur : true
```

### Auteur :

Jean-François Pech

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date création :

03/03/2023

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Convertir des objets en tableaux

Depuis sa version ES8 JS propose des fonctions utilisables sur les Objets qui vont pouvoir transformer leurs clés et ou leurs valeurs sous forme de tableau (pour utiliser les  $f^o$  forEach ou map par exemple).

```
//? Parcourir les Objet (Depuis javaScript ES8)
//? La Method .keys() qui convertit les clés d'un objet en tableau
//? La Method .values() qui convertit les valeurs d'un objet en tableau
//? La Method .entries() qui renvoie un tableau dans un tableau pour combiner clé - valeur
const keyUser = Object.keys(userData);
console.log("les clé de l'objet converties en array : ",keyUser);

const valuesUser = Object.values(userData);
console.log("les valeur de l'objet converties en array : ",valuesUser);

const convertedDataUser = Object.entries(userData);
console.log("les entrées de l'objet converties en array : ",convertedDataUser);
```

```
// De fait, une fois les objets convertis en tableau on peut ruser et utiliser forEach
par exemple :
valuesUser.forEach((lesValeurs)=>
    console.log(`FOREACH avec objet converti en tableau chaque valeurs : ${lesValeurs}
`);
);
convertedDataUser.forEach(([key, value])=>{
    console.log(`FOREACH avec objet converti en tableau : ${key}: ${value}`);
});
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : boucles

```
// TODO : JS map phase 1
// TODO : côté template html rajouter plein de <p></p>
// TODO : On va récupérer TOUS les <p> de notre page dans une
variable lesTxt via getElementsByTagName
// TODO : On va faire un console log de lesTxt
```

```
//TODO JS map Phase 2
//TODO Avec la méthode Array.from(), dans une nouvelle variable
textesTab on va transformer notre htmlCollection en array
//TODO On console log la variables textesTab
/* On transforme le HTMLCollection(qui contient tous nos <p>) en
Array classique
```

```
//TODO JS Map Phase 3 (on peut travailler sur un Array)
//TODO Sur textesTab on va utiliser la f° map(),
//TODO dans map(), on va lui passer en param une fonction fléchée
qui elle a en paramètre une variable temporaire
(nom de la variable au choix)
//TODO cette fonction fléchée elle va modifier le innerHTML ou
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution

```
const lesTxt = document.body.getElementsByTagName("p");
console.log(lesTxt);
```

```
/* On transforme le HTMLCollection (qui contient tous nos <p>) en
Array classique */
const textesTab = Array.from(lesTxt);
console.log(textesTab);
```

```
textesTab.map(uneCase => uneCase.innerHTML = "LOL JE SUIS
HACKERMAN" );
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

i <http://127.0.0.1:5500>

## Les système de boucles

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Totam et rem accusantium dicta voluptatum quam suscipit, molestiae dolor, nihil asperiores corrupti, quisquam ducimus! Sint, ipsa. Voluptates adipisci facilis veniam impedit?

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Totam et rem accusantium dicta voluptatum quam suscipit, molestiae dolor, nihil asperiores corrupti, quisquam ducimus! Sint, ipsa. Voluptates adipisci facilis veniam impedit?

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Totam et rem accusantium dicta voluptatum quam suscipit, molestiae dolor, nihil asperiores corrupti, quisquam ducimus! Sint, ipsa. Voluptates adipisci facilis veniam impedit?

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Totam et rem accusantium dicta voluptatum quam suscipit, molestiae dolor, nihil asperiores corrupti, quisquam ducimus! Sint, ipsa. Voluptates adipisci facilis veniam impedit?

**Auteur :**  
Jean-François Pech

Date création :

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

i <http://127.0.0.1:5500>

## Les système de boucles

LOL JE SUIS HACKERMAN  
LOL JE SUIS HACKERMAN

<https://github.com/jefff404/cours-js/tree/10-boucle>

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Spread Operator

Le spread operator est une fonctionnalité puissante de JavaScript qui permet de manipuler facilement des tableaux et des objets.

Le spread operator est représenté par trois points de suspension .... Il permet de décomposer un tableau ou un objet en éléments individuels. Cela signifie que vous pouvez accéder aux éléments d'un tableau ou aux propriétés d'un objet de manière plus concise et pratique.

Utilisation avec des tableaux

L'une des utilisations les plus courantes du spread operator est la copie d'un tableau existant. Plutôt que de créer une nouvelle référence pointant vers le même tableau, le spread operator crée une copie indépendante du tableau. Voici un exemple :

```
const tableauOriginal = [1, 2, 3];
const copieTableau = [...tableauOriginal];

console.log(tableauOriginal);
console.log(copieTableau);
```

Dans cet exemple, copieTableau est une copie exacte de tableauOriginal.

Le spread operator peut également être utilisé pour fusionner plusieurs tableaux en un seul.

```
///! Exemple pour faire une fusion
const tableau1 = [1, 2, 3];
const tableau2 = [4, 5, 6];
const tableauFusionne = [...tableau1, ...tableau2];
```

Maintenant, tableauFusionne contient tous les éléments des deux tableaux tableau1 et tableau2.

Utilisation avec des objets

En plus des tableaux, le spread operator peut également être utilisé avec des objets. Lorsqu'il est utilisé avec des objets, le spread operator copie toutes les propriétés de l'objet source dans un nouvel objet. Voici un exemple :

```
///! Test avec des objets
const objetSource = { a: 1, b: 2 };
const nouvelObjet = { ...objetSource };
console.log(nouvelObjet);
```

Maintenant, nouvelObjet contient les mêmes propriétés que objetSource.

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Le spread operator permet également de manipuler des éléments individuels d'un tableau. Vous pouvez ajouter de nouveaux éléments à un tableau existant ou modifier des éléments existants en utilisant le spread operator.

```
const tableauModifie = [...tableauOriginal, 4]; // Ajouter un nouvel élément à la fin du tableau

const tableauOriginal2 = [1, 2, 3];
const tableauModifie2 = [0, ...tableauOriginal2.slice(1)]; // Modifier le premier élément du tableau

const tableauOriginal3 = [1, 2, 3];
const tableauModifie3 =
[...tableauOriginal3.slice(0, 2), 10, ...tableauOriginal3.slice(2)];
// Remplacer un élément spécifique par un autre
```

<https://github.com/jeff404/cours-js/tree/11-spread-operator>

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Point sur var let ou const

En JavaScript de base pour déclarer une variable on utilise le mot « var », mais avec la version ES6 on a eu deux autres manières de déclarer des variables.

(Var cela permet de voir si un tutoriel commence à dater)

En fait var peut poser problèmes avec la notion de scope que l'on a vu précédemment

Rappel scope : Jusqu'où notre variable va être disponible

Avec let et const on gère le scope en fonction des blocs dans lesquels on se situe.  
(Scope de bloc)

C'est plus de contraintes mais ça permet de garder une logique et un code plus propre  
Avec sa version ES6 JS introduit des nouvelles manières de déclarer des variables

---

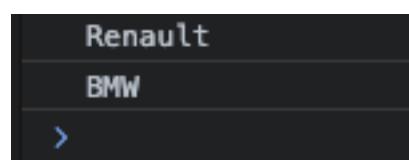
### Exercice : Quizz Var

Je suis stagiaire dans votre entreprise (sous traitant Nasa) et je vous envoi ce code en « revue de code »

Que me répondez-vous ?

```
var voiture = "Renault";
console.log(voiture);
var voiture = "BMW";
console.log(voiture);
```

(Je vous jure cela fonctionne)



#### Auteur :

Jean-François Pech

#### Date création :

03/03/2023

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Cela fonctionne mais ce n'est pas correct dans la logique.  
JS est peut être un langage assez permissif, mais cela peut engendrer des problèmes.

---

### Exercice : Quizz let

```
//!Quizz : ca bug
console.log(bolide);
let bolide = 'Jaguar'
```

```
✖ ▶ Uncaught ReferenceError: Cannot access 'bolide' before
initialization
    at app.js:12:13
(anonymous) @ app.js:12
```

---

### Exercice : Quizz function-var

```
function choixVoiture(){
    var uneVoiture = "Harley Davidson"
}

choixVoiture();
console.log(uneVoiture);
```

```
✖ ▶ Uncaught ReferenceError: uneVoiture is not defined
    at app.js:22:13
```

**Auteur :**  
Jean-François Pech  
**Relu, validé & visé par :**  
 Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date création :**

03/03/2023

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution fonction var

Pour le quizz précédent

Erreur : La var voiture est déclaré au sein de ma fonction et ne peut pas être utilisée en dehors.

```
var uneVoiture = "Harley Davidson"
function choixVoiture(){
}

choixVoiture();
console.log(uneVoiture);
```

Ici ça fonctionne car var est déclaré au-dessus.

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : Quizz if-var

```
var car = "Nissan";  
  
if(car=="Nissan"){  
    var vitesse = 800;  
}  
console.log(vitesse);
```

Ca fonctionne, pour vous est-ce normal ?

800

app.js:38

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Var est basé sur un scope de fonction uniquement  
If il ne le considère pas comme une fonction du coup ça fonctionne.

Si l'on y prête pas attention ce genre de soucis peut engendrer de plus gros problèmes quand le code de vos applications deviendra plus complexe

```
var theCar = "Nissan";

if(theCar=="Nissan"){
    let speed = 800;
}
console.log(speed);
```

✖ ▶ **Uncaught ReferenceError: speed is not defined** app.js:46  
at [app.js:46:13](#)

Le let fonctionne comme le var sauf que :  
il ne rend les variables disponible que à un bloc {}  
Boucle, fonction, condition peu importe.

Reprendons les Bug de tout à l'heure mais en utilisant le mot clé **let**

```
console.log(moto);
let moto = 'Yamaha'
```

✖ ▶ **Uncaught ReferenceError: Cannot access 'moto' before initialization** app.js:48  
at [app.js:48:13](#)

On obtient dès lors des erreurs certes mais beaucoup plus précises.

```
let voiture2 = 'Mitsubishi';
const modèle = 'Sport';

let voiture2 = 'Citroen';

✖ Uncaught SyntaxError: Identifier 'voiture2' has already been declared (at app.js:54:5)      app.js:54
```

Ici avec const : erreur voiture2 a déjà été déclarée

### Exercice Quizz : if-let

```
let superCar = 'BMW';
const superModel = 'Sport';

if(superCar =='BMW'){
    const superVitesse = 900;
    let superCar = "Citroen";
    console.log(superCar);

}
console.log(superCar);
```

#### Auteur :

Jean-François Pech

#### Date création :

03/03/2023

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution if-let

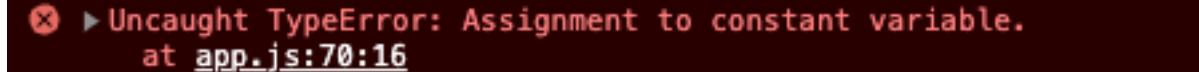
Ici c'est Ok Il faut bien comprendre que JS va créer 2 variable Différentes  
(le voiture du haut n'est pas le même qu'en bas)  
Importance du scope

## Const

```
const superConstante = 'YES';
superConstante = 12;
```

Une constante c'est une variable dont on sait qu'on ne va pas lui ré assigner une valeur, cela permet d'appliquer plus de restrictions dans notre code afin de ne pas créer plusieurs variable du même nom (des doublons) et qui servent à la même chose dans le programme.

Le code est mieux structuré donc plus lisible / compréhensible donc plus facile à maintenir.



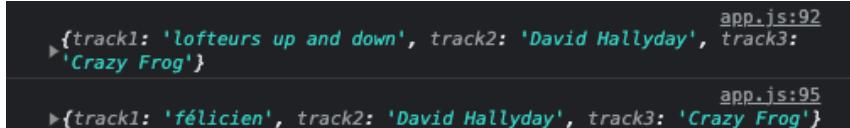
✖ ► Uncaught TypeError: Assignment to constant variable.  
at app.js:70:16

Une légère exception ?

On peut quand même « appliquer une modification » à une constante si c'est un objet, on peut modifier une **propriété** de cet objet (ci dessous on ne peut pas modifier la structure (l'objet en lui-même), mais seulement une de ses propriétés.

```
const MyTracklist = [
  track1: 'lofteurs up and down',
  track2: 'David Hallyday',
  track3: 'Crazy Frog'
]
console.log(MyTracklist);

MyTracklist.track1 = 'félicien'
```



```
▶ {track1: 'lofteurs up and down', track2: 'David Hallyday', track3: 'Crazy Frog'}  
app.js:92
▶ {track1: 'félicien', track2: 'David Hallyday', track3: 'Crazy Frog'}  
app.js:95
```

Au final en JS moderne (ES6 et +) on va privilégier l'utilisation de **let** et **const** pour créer des variables.

## Gestion des erreurs

### Try ... Catch

Aspect très important de la programmation d'autant plus quand on travaille dans une équipe, progressivement vos applications vont se complexifier et contenir beaucoup de code, d'objet, de fonctions etc..

Il y a des erreurs plus ou moins grave et de sources différentes

Les classiques :

- Erreur de Syntaxe, oubli d'un « ; », length ou lenght ?, etc...
- Erreur qui vient du serveur (pratique on peut accuser les développeurs BackEnd, l'incapacité à charger un fichier (404, etc...))
- Erreur qui vient du navigateur
- Erreur qui vient de l'utilisateur (envoyer une valeur non conforme dans un formulaire par exemple)

Dans la majorité des cas, une erreur va provoquer l'arrêt brutal d'un script et on risque donc d'avoir des codes et des pages non fonctionnelles.

Dans le pire des cas, une erreur peut être utilisée comme faille de sécurité par des utilisateurs malveillants qui vont l'utiliser pour dérober des informations ou pour tromper les autres visiteurs.

Javascript comporte nativement des fonctionnalités pour gérer les cas d'erreurs.

Dans tous les cas vous avez pu le constater, quand votre code comporte une erreur (vous avez un message d'erreur de base dans la console du navigateur), cela signifie que de base Javascript va créer, générer une erreur à partir de l'objet global Error (un objet de base dans javascript)

Par la suite nous allons pouvoir capturer l'objet d'Error renvoyé par Javascript et pouvoir indiquer ce que l'on souhaite faire dans le code quand cette erreur survient.

On va avoir à disposition un syntaxe de bloc try ... catch....

Dans le bloc try : on écrit le code à tester (qui peut potentiellement générer des erreurs)

Dans le bloc catch : on écrit le code pour gérer l'erreur (on fait un simple console.log ? On affiche un message à l'utilisateur ? On enregistre quelque chose en base de données ?

Nous allons voir le gestion d'erreurs au travers d'exemples.

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

```
prenom;  
alert('Ce message ne s\'affichera pas');
```

Erreur généré par la fonction alerte

 **Uncaught SyntaxError: Invalid or unexpected token (at app.js:4:7)**      [app.js:4](#)

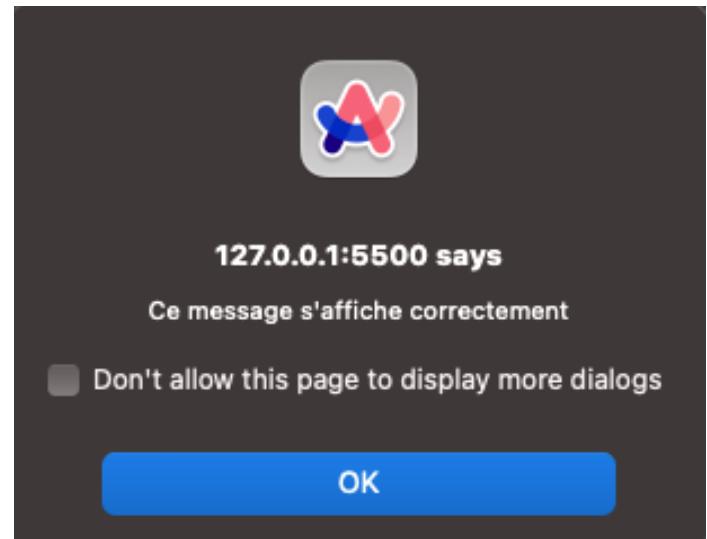
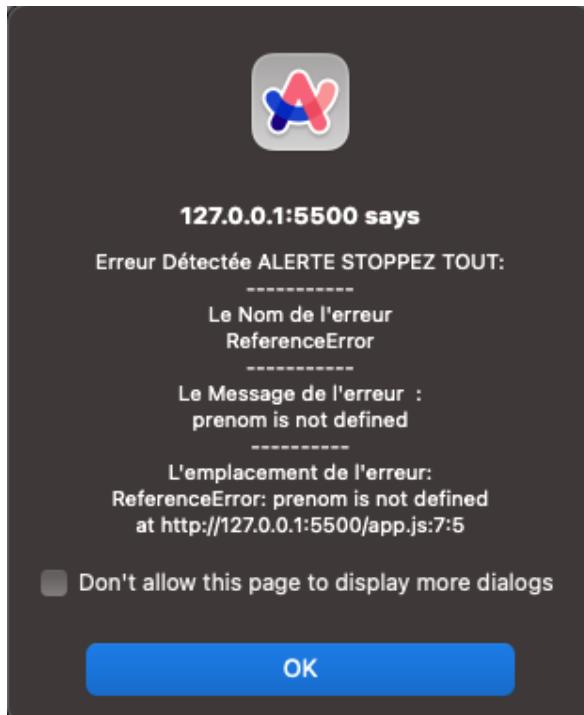
Erreur généré par la variable prénom

 **▶ Uncaught ReferenceError: prenom is not defined**      [app.js:3](#)  
at [app.js:3:1](#)

Donc si on sait que ce code d'exemple peut nous créer des erreurs on peut l'écrire de cette manière :

```
try{  
    prenom  
    alert('Bonjour');  
}catch(err){  
    alert(`Erreur DéTECTée ALERTE STOPPEZ TOUT:  
-----  
Le Nom de l'erreur :  
${err.name}  
-----  
Le Message de l'erreur :  
${err.message}  
-----  
L'emplacement de l'erreur:  
${err.stack}`);  
}  
alert(`Ce message s'affiche correctement`);
```

Comme on a un erreur dans le code du bloc Try seule les alertes du bloc catch puis ensuite la dernière alert sera affichée à l'utilisateur.



## Gestion des exceptions avec Throw

Dans certains cas, lorsque l'on écrit les différentes fonctions d'un programme, il se peut que le code soit juste mais cela ne correspond plus à la réalité.

Imaginez vous réalisez une application pour gérer des retrait et virement bancaires, vous avez 10 euros sur votre compte et vous voulez faire un retrait de 50 K€, techniquement si on se place du point de vue du code il nous faudra certainement une fonction qui fait une soustraction.

Donc du point de vue du code on peut faire 10-50000, ça va fonctionner et votre solde sera de -49990 €.

Le code est juste d'un point de vue technique mais du point de vue du Banquier peut être pas.

Dans notre cas il faut bien prendre en compte les règles de gestion de l'application, et donc ici il nous faudra mettre en place une exception dans le cas où le montant que voudrait retiré serait supérieur au solde de l'utilisateur.

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Ici ce programme a pour but de demander 2 données à l'utilisateur (la fonction prompt), pour ensuite diviser ces données.

On va donc mettre en place 2 exceptions (il pourrait y en avoir plus )

1 exception si l'utilisateur ne renseigne pas 2 nombres.

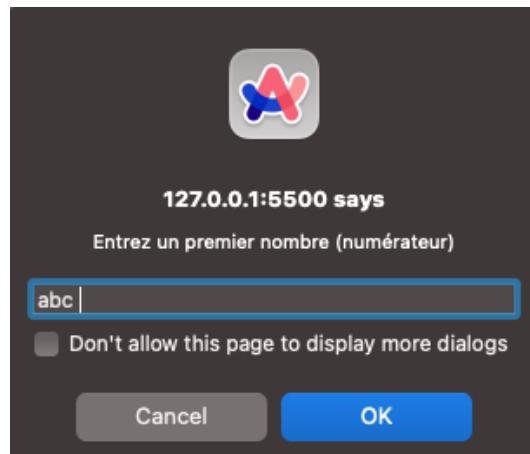
1 exception si l'utilisateur essaye de nous faire diviser par zéro

```
function division(){
    let x = prompt('Entrez un premier nombre (numérateur)');
    let y = prompt('Entrez un deuxième nombre (dénominateur)');

    if(isNaN(x) || isNaN(y) || x == '' || y == ''){
        throw new Error('Merci de rentrer deux nombres');
    }else if(y == 0){
        throw new Error('Division par 0 impossible')
    }else{
        alert(x / y);
    }
}

division();
```

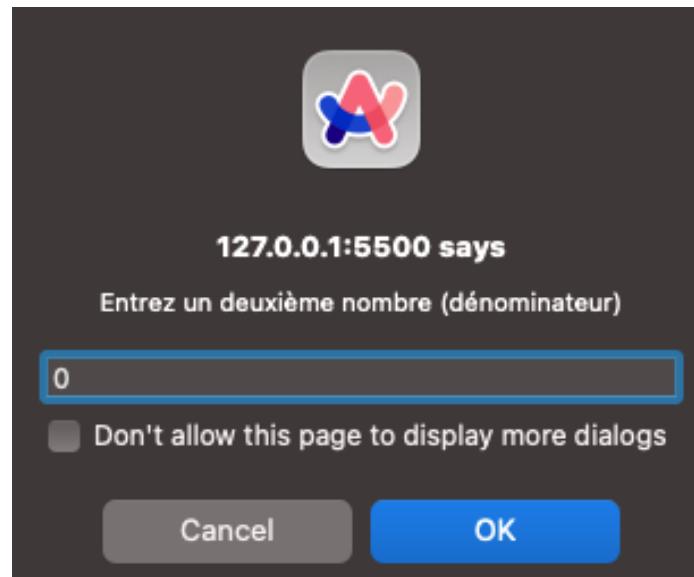
1 ère exception :



```
✖ ▶ Uncaught Error: Merci de rentrer deux nombres
  at div (app.js:29:15)
  at app.js:37:1
```

[app.js:29](#)

2ème exception :



✖ ► Uncaught Error: Division par 0 impossible  
at div ([app.js:31:15](#))  
at [app.js:37:1](#)

app.js:31

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

✓ Jérôme CHRETIENNE  
✓ Sophie POULAKOS  
✓ Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Finally

Au sein d'un bloc try catch, on peut (optionnel) utiliser l'instruction Finally, ce bloc nous permet de préciser du code qui sera exécuté dans tous les cas, qu'une erreur ou exception ait été générée ou pas.

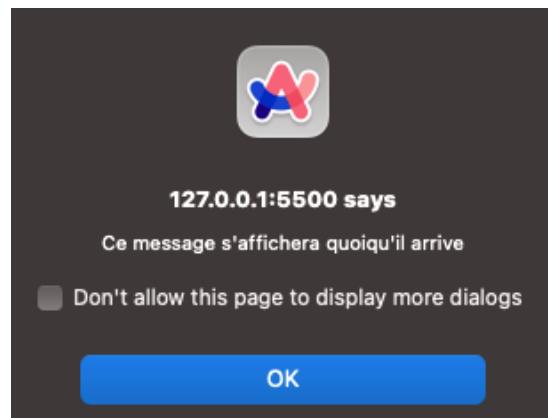
Reprendons notre fonction division : ici on ne va plus exécuter la fonction division directement dans notre programme on va **essayer** d'exécuter la fonction.

```
function division(){
    let x = prompt('Entrez un premier nombre (numérateur)');
    let y = prompt('Entrez un deuxième nombre (dénominateur)');

    if(isNaN(x) || isNaN(y) || x == '' || y == ''){
        throw new Error('Merci de rentrer deux nombres');
    }else if(y == 0){
        throw new Error('Division par 0 impossible')
    }else{
        alert(x / y);
    }
}

// division();

try{
    division();
}catch(err){
    alert(err.message);
}finally{
    alert(`Ce message s'affichera quoiqu'il arrive`);
}
```



## Les Classes

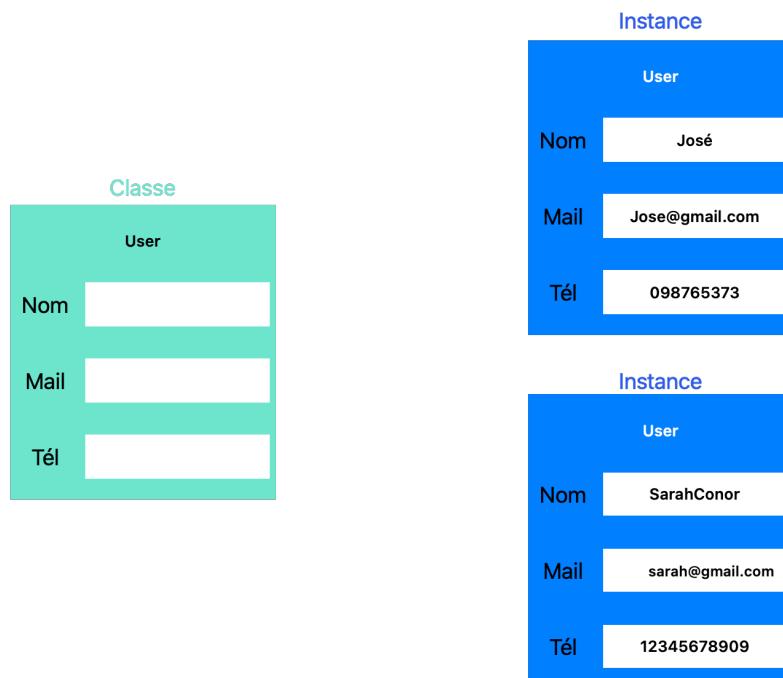
Pré requis : Les objets

Les classes sont un élément essentiel de le P.O.O (Programmation orientée objet), pour résumer le concept les classes vont nous permettre de créer plus facilement et rapidement plusieurs objets avec des caractéristiques, d'architectures similaires.

Par exemple si une application gère des utilisateurs on peut définir une classe « user » et pour chaque user on gère les données suivantes : un nom, un mail, un num de téléphone.

Pour créer / **construire** des nouveaux user, le système de class utilise une fonction qui s'appelle **constructor (construct** dans d'autres langages)

On va pouvoir plus facilement créer de nouveaux user (des nouvelles instances de la classe user)



Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Syntaxe

Donc côté code, on crée notre class UserProfile, pour pouvoir créer des nouvelles instance on renseigne les données dont on a besoin, que l'on recevra en paramètre (nameUser, mailUser, phoneUser)

Le mot clé this va représenter l'objet courant, celui que l'on est entrain de créer, c'est le contexte. Ici pour résumer on assigne aux propriété de notre classe les valeur qu'on va recevoir en paramètre

```
class UserProfile {
    constructor(nameUser, mailUser, phoneUser) {
        this.nameUser = nameUser;
        this.mailUser = mailUser;
        this.phoneUser = phoneUser;
    }
    getProfileInfo() {
        return `infos de l'utilisateur :
            son nom : ${this.nameUser}
            son mail : ${this.mailUser}
            son Tél : ${this.phoneUser}`;
    }
}
```

Bonus : on a écrit une fonction au sein de la classe, c'est une méthode de classe et elle ne pourra s'utiliser QUE sur des objets (des nouvelles instances) de cette classe

Une fois qu'on a définit la structure de notre classe on va pouvoir utiliser le constructeur pour créer un nouvel utilisateur en faisant new UserProfile()

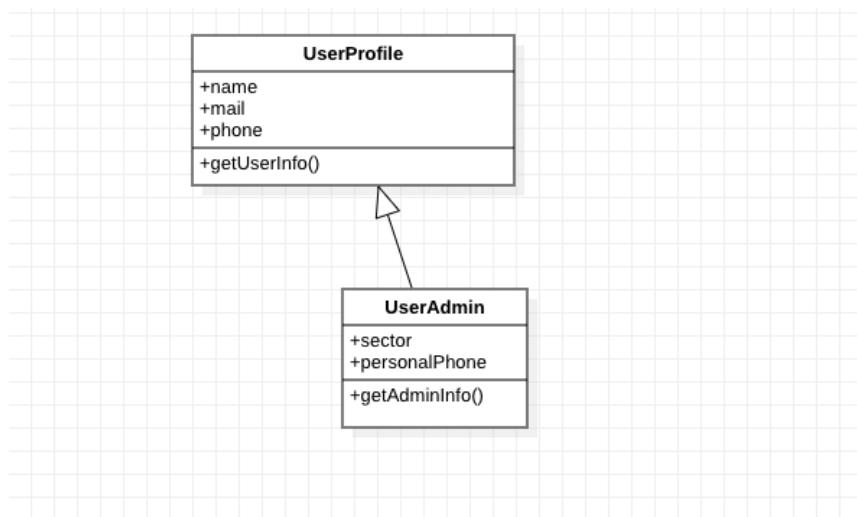
```
const exampleUser1 = new UserProfile("José", "jose@gmail.com",
"09876543");
```

```
const exampleUser2 = new UserProfile("Sarah", "sarah@gmail.com",
"063736252");
exampleUser2.getProfileInfo();
```

Dans notre cas Il n'y aura que sur des new UserProfile que l'on pourra utiliser la méthode getProfileInfo().

## L'héritage

Avec les classes on peut également profiter d'un système d'héritage, cela signifie que nous pouvons étendre (**extends**) les propriétés, les méthodes d'une classe vers une autre, Par exemple dans notre application on gère déjà des utilisateurs, mais on veut aussi gérer des utilisateurs un peu plus spécifiques : des Admin, les admins ils auraient les mêmes propriétés que les utilisateurs (un nom, un mail, un téléphone) mais avec des informations en plus (le **secteur** dans lequel l'admin travaille, et son **Téléphone personnel**) on crée une nouvelle classe « enfant » qui hérite des propriétés et des méthodes d'une classe parent.



⚠ une classe enfant peut hériter d'une classe parent mais l'inverse n'est pas possible.  
Sur une instance de **UserAdmin** on pourra utiliser **getProfileInfo()** mais sur une instance de **UserProfile** on ne peut pas utiliser **getAdminInfo()**.

### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Dans le code : on va utiliser **extends** et **super()**

```

class UserProfile {
    //! Pas besoin de déclarer function devant le constructor et méthodes
    constructor(nameUser, mailUser, phoneUser) {
        this.nameUser = nameUser;
        this.mailUser = mailUser;
        this.phoneUser = phoneUser;
    }
    getProfileInfo() {
        return `infos de l'utilisateur :
            son nom : ${this.nameUser}
            son mail : ${this.mailUser}
            son Tél : ${this.phoneUser}`;
    }
}

const exampleUser1 = new UserProfile("José", "jose@gmail.com", "09876543");
const exampleUser2 = new UserProfile("Sarah", "sarah@gmail.com", "063736252");
exampleUser2.getProfileInfo();

class UserAdmin extends UserProfile{
    constructor(unNom,unMail,unPhone,sector,personnalPhone){
        super(unNom,unMail,unPhone); //! Appel au constructor du parent
        this.sector = sector;
        this.personnalPhone = personnalPhone;
    }
    getAdminInfo(){
        return `infos de l'utilisateur :
            son nom : ${this.nameUser}
            son secteur d'intervention : ${this.sector}
            son Tél Personnel : ${this.personnalPhone}`;
    }
}

const exampleAdmin1 = new UserAdmin('Jacky','jack@gmail.com','012345678','administration','0987654323');

console.log(exampleAdmin1.getAdminInfo());

```

#### Auteur :

Jean-François Pech

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date création :

03/03/2023

#### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exo Class IMC

Créer un programme permettant de Calculer l'IMC d'une personne

TODO :

- Créer une classe Imc avec un constructeur qui recevra un nom, un poids, une taille
- Créer une fonction de calcul d'IMC, qui retourne le résultat du calcul (à 2 nombre après la virgule si possible)
- Créer une fonction d'affichage « display », elle a pour rôle d'afficher en console :  
Le nom de la personne, son poids, sa taille et son imc dans une phrase
- En dehors de la class (donc dans le programme principal), récupérer la variable list (un tableau de nouvelle instances de la class) (voir discord ou 
- Trouver un moyen de parcourir les éléments dans la variable list, sur chaque element utiliser la fonction display

En console du navigateur :

```
Sébastien Chabal (135 kg, 1.7 M) a un IMC de: 46.71
Escaladeuse (45 kg, 1.68 M) a un IMC de: 15.94
JOJO (300 kg, 2 M) a un IMC de: 75.00
Gontrand (90 kg, 1.75 M) a un IMC de: 29.39
Colonel Clock (200 kg, 1.75 M) a un IMC de: 65.31
J0siane de la Vega (99 kg, 1.55 M) a un IMC de: 41.21
```

Programme Principal (en dehors de la classe)

```
// /* progr principal -> on fait l'injection des données
let list = [
    new Imc("Sébastien Chabal", 135, 1.7),
    new Imc("Escaladeuse", 45, 1.68),
    new Imc("JOJO ", 300, 2),
    new Imc("Gontrand ", 90, 1.75),
    new Imc("Colonel Clock ", 200, 1.75),
    new Imc("J0siane de la Vega", 99, 1.55),
];
/*Boucle qui parcourt list pour utiliser display()
????.????????((??????) ?? ????.????());
```

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exo Class PME

Gérer une PME

CDC

Un Salarié a un nom, prénom, âge, salaire mensuel  
Il est payé sur N mois.  
En plus il y a XXX de charges

Une Pme c'est un nom, une équipe de plusieurs salariés  
Grâce à ses ventes elle a des revenus R  
Mais aussi ... :  

- des frais fixes FF (impôts etc...)
- Des frais d'achats de matériel et de logiciels FA

TODO :

- Créer une classe Pme et une classe Employee
- Utiliser des fonctions
- Faire le bilan annuel de l'entreprise et l'afficher en console.

Détails :

- 3 salariés qui gagnent par mois : 2000, 3000 et 4000 euros
- R = 300000 (trois cent mille)
- FF = 20000 (vingt mille)
- FA = 50000 (cinquante mille)
- N = 12
- XXX = 90%

En console du navigateur :

```
-----MA PME-----
Ma Petite Entreprise - : Cout Initial : 70000
Ma Petite Entreprise - : Cout Total Equipe : 205200
Ma Petite Entreprise - : VENTES : 300000
Ma Petite Entreprise - : BILAN : 24800
```

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

```
// // Scénario
const pme = new Pme (
    //Le nom entreprise
    "Ma Petite Entreprise -",
    //L'équipe de salariés (un tableau)
    [new Employee ("Duval", "Paul", 30, 2000),
     new Employee ("Durand", "Alain", 40, 3000),
     new Employee ("Dois", "Sylvia", 50, 4000)],
    //le revenu , frais fixe, frais d'achat
    300000,
    20000,
    50000);
pme.bilanCalculed();
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exo Class COMPTES BANCAIRES.

### Enoncé

Gérer des compte en banque

### Consignes

- Créer une classe CompteBancaire avec des méthodes de crédit, de retrait, de visualisation de l'état du compte bancaire (en console), on doit pouvoir aussi faire un virement d'un membre à un autre.
- Générer une exception pour ne pas dépasser le solde (pas de retrait ou de virement qui dépassent le solde du compte bancaire)

### Détails

Faire une scénario avec gestion de 3 comptes crédités à 1000 € chacun (Alex, Clovis, Marco)  
Puis Alex retire 100

Puis Marco fait un virement de 300 à Clovis

Enfin Alex tente un retrait de 1200

Afficher tous les soldes finaux.

Ces compte sont placés dans un tableau associatif de clients

En conso

```
Ajout de: 1000 pour: Alex
Ajout de: 1000 pour: Clovis
Ajout de: 1000 pour: Marco
Retrait de: 100 pour: Alex
Virement de: 300 de: Marco vers: Clovis
Ajout de: 300 pour: Clovis
Retrait de: 300 pour: Marco
----->Alex, retrait de: 1200 refusé avec solde: 900
titulaire: Alex, solde: 900
titulaire: Clovis, solde: 1300
titulaire: Marco, solde: 700
```

Des ressources :

[https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Working\\_with\\_objects](https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Working_with_objects)

<https://www.pierre-giraud.com/javascript-apprendre-coder-cours/gestion-erreur-exception-try-catch/>

<https://www.pierre-giraud.com/javascript-apprendre-coder-cours/gestion-erreur-exception-try-catch/>

<https://javascript.info/try-catch>

#### Auteur :

Jean-François Pech

#### Date création :

03/03/2023

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

```
// Main, gère 3 comptes bancaires dans un tableau associatif
const lesComptes = {
  Alex: new CompteBancaire("Alex"),
  Clovis: new CompteBancaire("Clovis"),
  Marco: new CompteBancaire("Marco"),
};

// lecture tableau associatif ou Objet["truc"]
// Crédite et décrète chaque compte
for (let key in lesComptes) lesComptes[key].crediter(1000);

// un retrait de 100 par Alex
??????????????;
// un petit virement: Marco Vire 300 à Clovis
??????????????;
// un petit retrait incorrect (doit déclencher erreur custom) :
// Alex fait un retrait de 1200
?????;
// bilan : faire une description de tous les comptes en console (ou DOM ?)
?????;
```

## La guerre des langages

☒ En programmation on peut distinguer à peu près tous les langage en 2 familles, les langages basés sur les **classes** et ceux basés sur les **prototypes**.  
Js est un langage orienté objet basé sur les prototypes. (C'est pour cela que l'on dit qu'en Javascript TOUT est Objet)

Le JavaScript est un langage objet basé sur les prototypes. Cela signifie que le JavaScript ne possède qu'un type d'élément : **les objets** et que tout objet va pouvoir partager ses propriétés avec un autre, c'est-à-dire servir de prototype pour de nouveaux objets.  
L'héritage en JavaScript se fait en remontant la chaîne de prototypage.

En plus de la manière déclarative (créer un variable et lui assigner directement une valeur) dans Javascript on va retrouver également un système de constructor pour créer tout type d'objet

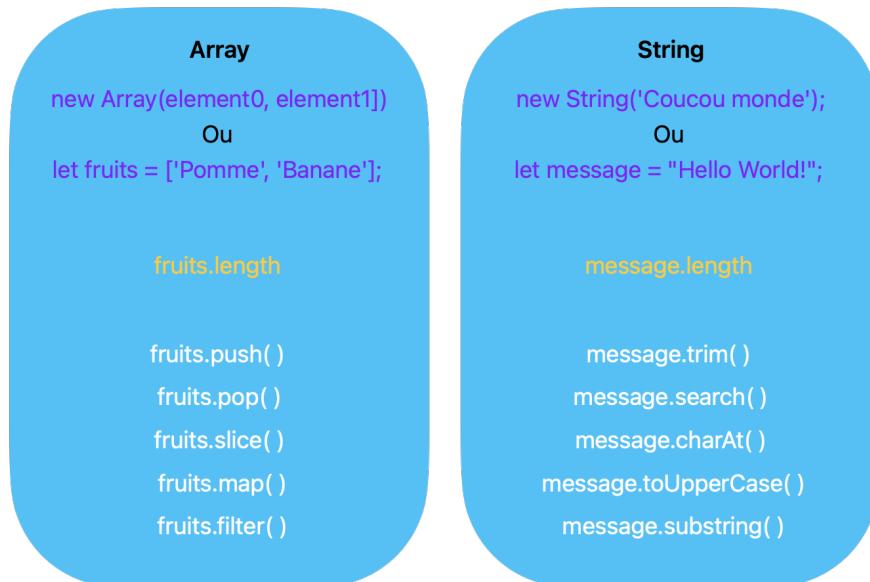
Exemple de fiche récapitulative des types d'objets en JS (non exhaustif) :

On peut déclarer soit en utilisant le constructor new Array( ), mais il faut penser à stocker dans une variable, nativement dans JS pour chaque objet on aura des propriétés de base, ici length commun à plusieurs type et JS propose aussi des fonctions de base, utiles pour manipuler chaque type d'objets. (Toujours avoir le réflexe d'aller consulter la documentation, NE PAS réinventer la ROUE)

Violet : créer une variable (via constructor ou en mode déclaratif)

Jaune : exemple d'une propriété

Blanc : des fonctions de bases



### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Asynchrone

### API

Une API (Application Programming Interface ou Interface de Programmation Applicative en français) est une interface, c'est-à-dire un ensemble de codes grâce auxquels un logiciel fournit des services à des clients.

Le principe et l'intérêt principal d'une API est de permettre à des personnes externes de pouvoir réaliser des opérations complexes et cachant justement cette complexité.

En effet, en tant que développeur nous n'aurons pas besoin de connaître les détails de la logique interne du logiciel tiers et n'y aura d'ailleurs pas accès directement puisque nous devrons justement passer par l'API qui va nous fournir en JavaScript un ensemble d'objets et donc de propriétés et de méthodes prêtes à l'emploi et nous permettant de réaliser des opérations complexes.

Il existe des API pour à peu près tout, les plus classiques que vous connaissez sont par exemple les API Google Maps, la météo, l'API Geolocation qui va nous permettre de définir des données de géolocalisation ou encore l'API Canvas qui permet de dessiner et de manipuler des graphiques dans une page.

Nous allons donc organiser notre code de manière à pouvoir contacter une API qui va nous renvoyer une réponse contenant des données, il existe plusieurs types d'API qui renvoi plusieurs types de réponse.

Actuellement la plupart des API sont des API restful c'est-à-dire que le format des réponses que renvoi l'API peut être en JSON, HTML, XLT, Python, PHP ou simplement du texte brut.

Désormais, beaucoup d'API vont renvoyer des réponses au format JSON (Javascript Object Notation), une notation d'objet en Javascript. (il existe une méthode native de JS pour transformer des objets JSON, la méthode .json())

Dans les faits techniquement pour nous une API ça sera simplement une URL que l'on contactera via Javascript pour en extraire les informations.

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Fetch()

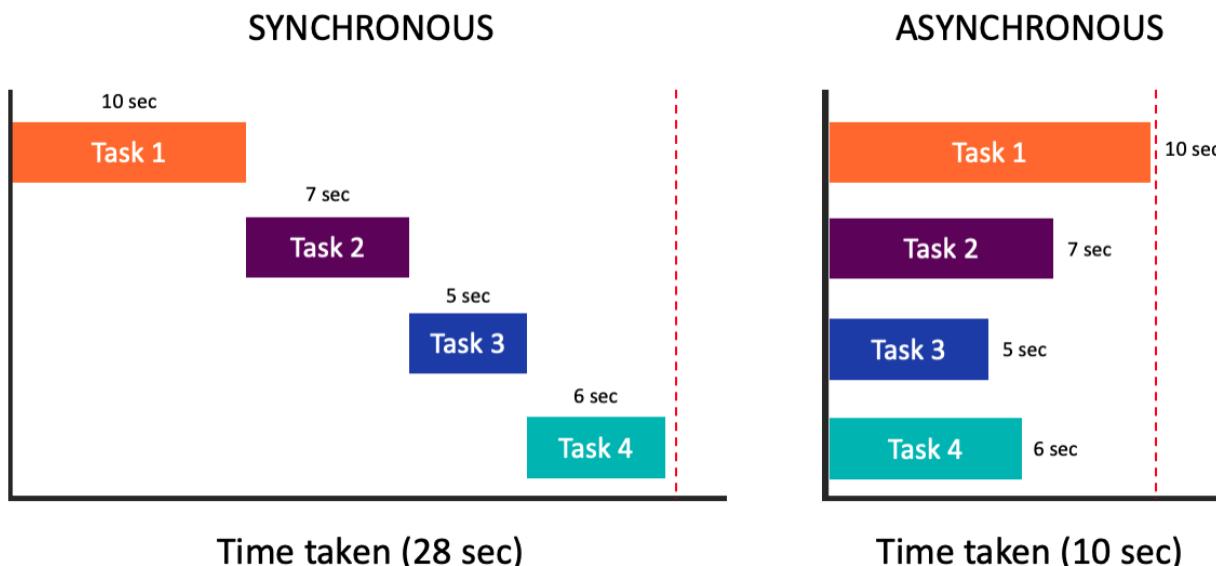
Dans Javascript, nous allons utiliser la méthode `fetch()`, qui nous permettra de contacter n'importe quelle API via son URL, la méthode renvoi des objets de type `Response` ou `Promise`.  
 l'API Fetch et sa méthode `fetch()` qui correspondent à la "nouvelle façon" d'effectuer des requêtes HTTP.

## Code Asynchrone

Un autre concept essentiel lorsque nous allons contacter des API, c'est d'organiser notre code de manière asynchrone.

L'idée c'est que de base nos programme Javascript sont en mode synchrone, à savoir que chaque ligne de code qui représente une instruction, celle-ci se termine ET seulement ensuite JS passe à l'exécution de la ligne de code suivante.

À la différence d'un code asynchrone qui va permettre d'exécuter (ou finir d'exécuter une ou plusieurs instructions) en parallèle.



### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Async await

On va pouvoir indiquer à javascript sur certaines instructions d'attendre que celle ci soit complétée avant de passer à l'instruction suivante.

On déclare une fonction avec le mot clé **async** pour indiquer que le code contenu dans cette fonction devra être exécuté de manière asynchrone et sur certaines instructions de cette fonction utiliser le mot clé **await** (généralement sur la fonction `fetch()` et la fonction `json()` qui manipule les données de l'api)

Exemple ci dessous on contacte une url d'api météo pour afficher la latitude dans la page web : Pour cet exemple on fait une fonction fléchée anonyme stocké dans une variable (pour s'habituer à ce genre de syntaxe), on précise avant la fonction qu'elle est en mode **async** puis quand on contact l'api via `fetch()` on précise que cette instruction est en mode **await** elle doit se finir totalement pour continuer la suite du programme, et quand on transforme la réponse de l'api avec la fonction `json()`, afin de transformer la réponse en objet javascript (plus facile à manipuler), idem on précise que c'est en mode **await**.

```
const apiDiv = document.querySelector('.apiContact');
//de base une f° => est anonyme, astuce pour désanonymiser, on la stocke dans une
variable
const contactApi = async () => {
    //Data va récup Toutes les données de l'api
    const data = await fetch('https://api.open-meteo.com/v1/forecast?
latitude=52.52&longitude=13.41&hourly=temperature_2m');
    console.log(data);
    //Plutôt que de Travailler sur la réponse, on va la transformé pour
    //qu'elle devient un OBJET JS (+ pratique)
    const dataTransformed = await data.json();
    console.log(dataTransformed);
    apiDiv.innerText = dataTransformed.latitude;
};
contactApi();
```

Prenez le réflexe de toujours se référer à la documentation officielle des API

### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Ci dessous le console log de data (les données brutes) que renvoie l'api

```
Response {type: 'cors', url: 'https://api.open-meteo.com/v1/forecast?latitude=52.52&longitude=13.41&hourly=temperature_2m'}
  ▼ lse, status: 200, ok: true, ...
    body: (...)

  ▶ headers: Headers {}
  ok: true
  redirected: false
  status: 200
  statusText: ""
  type: "cors"
  url: "https://api.open-meteo.com/v1/forecast?latitude=52.52&longitude=13.41&hourly=temperature_2m"
  ▶ [[Prototype]]: Response
```

C'est un objet de type **Response** qui contient plusieurs propriétés comme ok ou status : 200 (le contact avec l'api s'est bien passé)

On peut aussi gérer cela en JS pour des cas d'erreurs...

Le second console log correspond au données transformée en objet JS via la fonction `.json()`  
C'est là que l'on peut voir les données fournit par l'api, une fois transformée on accède aux données comme en mode objet

`dataTransformed.latitude`

```
app.js:40
  ▼ {latitude: 52.52, longitude: 13.419998, generationtime_ms: 0.35691261291503906, utc_offset_second
    ▼ s: 0, timezone: 'GMT', ...} ⓘ
      elevation: 38
      generationtime_ms: 0.35691261291503906
    ▶ hourly: {time: Array(168), temperature_2m: Array(168)}
    ▶ hourly_units: {time: 'iso8601', temperature_2m: '°C'}
      latitude: 52.52
      longitude: 13.419998
      timezone: "GMT"
      timezone_abbreviation: "GMT"
      utc_offset_seconds: 0
    ▶ [[Prototype]]: Object
```

**Auteur :**  
Jean-François Pech  
**Relu, validé & visé par :**  
Jérôme CHRETIENNE  
Sophie POULAKOS  
Mathieu PARIS

**Date création :**

03/03/2023

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : Contacter une API

Avec ce EndPoint d'api

<https://pokeapi.co/api/v2/pokemon>

Affichez dans une page web le nom des 20 premiers Pokemon

## Then catch

Autre manière de gérer le code de manière asynchrone avec le chainage à la fonction `fetch()`, des fonction(s) `then()` et enfin la fonction `catch()` pour capter et gérer les erreurs

Si on adapte cette méthode à notre premier exemple d'API (celle de la météo) :

```
// //** MÉTHODE avec Fetch + .then() + catch() */
const apiDiv = document.querySelector('.apiContact');
console.log(apiDiv);
const contactApi = () => {
    fetch('https://api.open-meteo.com/v1/forecast?
latitude=52.52&longitude=13.41&hourly=temperature_2m')
        .then(response => response.json())
        .then(data =>(apiDiv.innerText = data.latitude))
        .then(data =>(console.log(data)))
        .catch(error => console.log("Erreur custom : " + error));
}; contactApi();
```

# Fetch API

52.52

#### Auteur :

Jean-François Pech

#### Date création :

03/03/2023

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Gestion des erreurs Response ET Promise

Pour aller encore plus loin on peut également, en plus des erreurs de la Promise, via JS gérer les erreurs également au niveau de la response en accédant au propriétés de l'objet rappelez vous : Ci dessous le code permet de gérer dès la response une erreur si l'url du fetch est invalide. C'est mieux pour travailler en collaboratif et assurer un aspect **qualitatif** de votre code

```
▼ Response {type: 'cors', url: 'https://api.open-meteo.com/v1/forecast?latitude=52.52&longitude=13.41&hourly=temperature_2m', status: 200, ok: true, ...} ⓘ
  body: (...)  

  bodyUsed: true
▶ headers: Headers {}
  ok: true
  redirected: false
  status: 200
  statusText: ""
  type: "cors"
  url: "https://api.open-meteo.com/v1/forecast?latitude=52.52&longitude=13.41&hourly=temperature_2m"
▶ [[Prototype]]: Response
```

```
/** METHODE avec Fetch +then + catch + async Await */
const apiDiv = document.getElementById("apiContact");
const contactApi = () => {
    //! tester si jamais on se trompe dans l'url (mettre l'un des 2 fetch en commentaire)
    fetch("https://api.npms.io/on-s-est-trompe-dans-l-url")
    //! Ci dessous avec une url valide
    // fetch("https://api.open-meteo.com/v1/forecast?
    latitude=52.52&longitude=13.41&hourly=temperature_2m")
        .then(async (response) => {
            const dataTransformed = await response.json();
            // Ici on gère aussi les erreurs au niveau de la réponse de l'api
            // Si dans la réponse la propriété ok n'est pas définie
            if (!response.ok) {
                // on récupère les messages d'erreur ou la propriété statusText par default
                const error = (dataTransformed && dataTransformed.message) || response.statusText;
                //f° native de JS utilisé sur les objets de type Promise
                return Promise.reject(error);
            }
            apiDiv.innerText = dataTransformed.latitude;
        })
        .catch((error) => {
            console.log(error);
            // Ici on crée une error custom et l'objet error
            console.error("Attention une fusée à décollée depuis Grenoble", error);
        });
}
contactApi();
```

### Auteur :

Jean-François Pech

### Relu, validé & visé par :

Jérôme CHRETIENNE  
Sophie POULAKOS  
Mathieu PARIS

### Date création :

03/03/2023

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Javascript Modulaire

Import export  
Type module

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Web Workers

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023

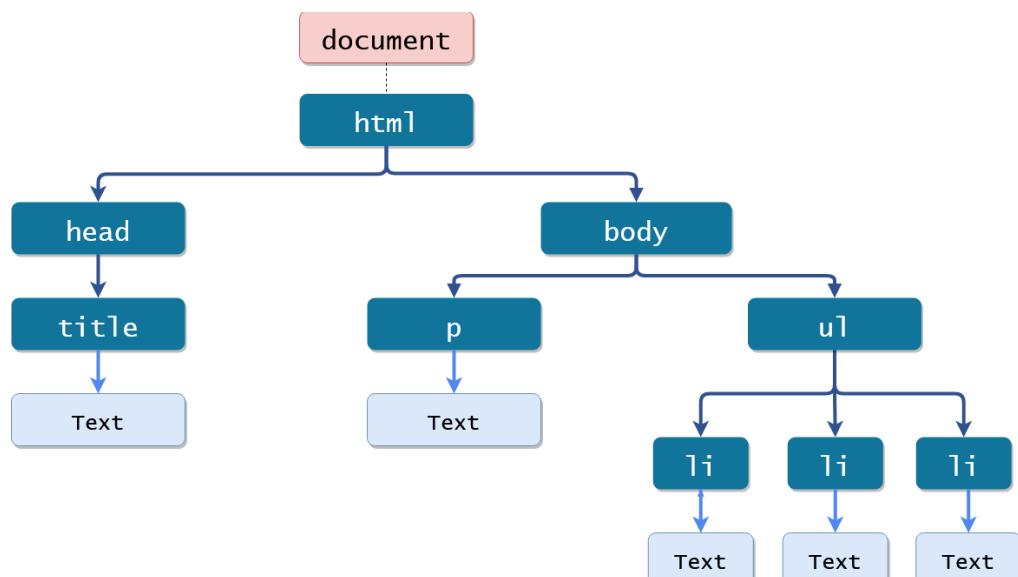


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## D.O.M

### Présentation

Le DOM est donc une arborescence que va fabriquer le navigateur quand il interprète un fichier HTML, comme cité précédemment en JS tout est objet, et dans le DOM nous allons pouvoir retrouver tous les éléments HTML de notre page sous forme d'objets (ayant des propriétés) manipulables par JS.



Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Sélectionner des éléments du DOM

### getElement(s)By

Une première manière de sélectionner un élément Html (un lien, un titre, une image, ... , n'importe quel élément Html).

#### getElementsByName() - getElementsByTagName()

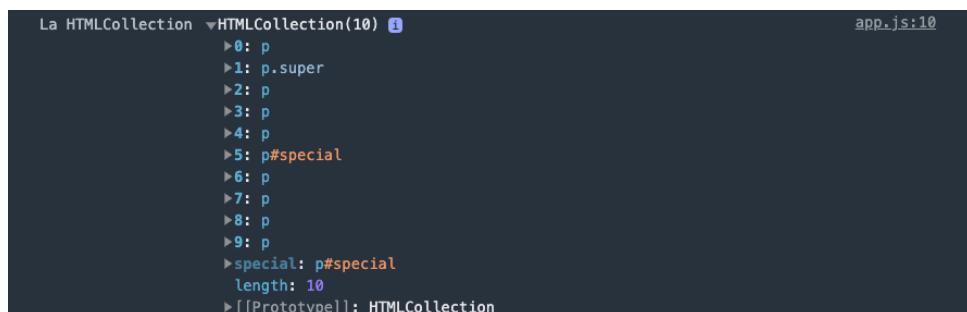
permet de sélectionner TOUS les éléments par le nom de leur balise (leur tag) html.

```
//? Une fonction type getElement pour récupérer tous les élément selon une certaine
balise dans une HTMLCollection
let tousLesP = document.getElementsByTagName('p');
console.log('La HTMLCollection',tousLesP);
//? Quand on a une HTMLCollection on peut accéder à un certains éléments
console.log('le 3e <p> dans la HTMLCollection : ',tousLesP[2]);
//? Une fonction type getElement pour récupérer tous les élément selon une certaine
class dans une HTMLCollection
let tousLesSuper = document.getElementsByClassName('super');
console.log(tousLesSuper);
```

La console du navigateur nous affiche un tableau de type HTMLCollection, un tableau qui va contenir tous les paragraphe <p> de notre page. On a accès à toutes les propriétés de ces éléments Html.

(Son contenu, son alignement, la couleur du texte, sa position dans la page, etc... )  
On constate également que ce tableau est indexé (à l'indice [0], le premier paragraphe <p> de la page)

**Défi :** dans la console, trouver les propriétés dans lesquelles on retrouve ce que l'on a écrit dans les paragraphes.



The screenshot shows a browser's developer tools console output. It displays an object named 'La HTMLCollection' which is identified as an 'HTMLCollection(10)'. The object has several properties listed:

- length: 10
- [Prototype]: HTMLCollection
- special: p#special
- 0: p
- 1: p.super
- 2: p
- 3: p
- 4: p
- 5: p#special
- 6: p
- 7: p
- 8: p
- 9: p

<b>Auteur :</b>	Jean-François Pech
<b>Relu, validé &amp; visé par :</b>	<input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> Mathieu PARIS

<b>Date création :</b>	03/03/2023
<b>Date révision :</b>	10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

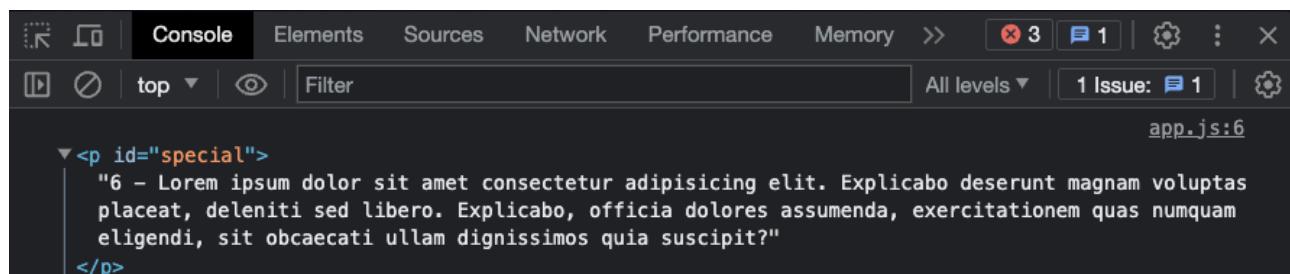
## getElementById()

On peut aussi sélectionner un élément html par son id, on va utiliser getElementById () (récupérer un élément selon le nom de son attribut id)

getElementById () renvoi directement les données elles-mêmes et non pas sous forme de HTMLCollection.

Il n'y a pas de S à Element dans le nom de la fonction, la console nous affichera le code html d'un seul élément en particulier (\*en html on n'a qu'un seul id avec un certain nom par page)

```
//? Une fonction type getElement pour récupérer UN élément par son ID
let specialP = document.getElementById('special');
console.log(specialP);
```



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The command `let specialP = document.getElementById('special');` is entered, followed by `console.log(specialP);`. The output in the console is:

```
<p id="special">
  "6 – Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas
  placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam
  eligendi, sit obcaecati ullam dignissimos quia suscipit?"</p>
```

The file path `app.js:6` is shown at the bottom right of the log entry.

### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## QuerySelector( )

Une alternative, les fonctions de type querySelector, qui vont se basées sur la syntaxe de css (rappel : Les selector en CSS)

### querySelector()

```
//? Une fonction type querySelector pour récupérer UN élément (le 1er trouvé)
let lePremierP = document.querySelector('p');
console.log('lePremierP via querySelector : ', lePremierP);
```

 sélection par l'id (avec # comme en css)

```
//? Une fonction type querySelector pour récupérer UN élément par son ID
let pSpecial = document.querySelector('#special');
console.log('pSpecial querySelector + ID',pSpecial);
```

 Sélection par la classe (avec . comme en css)

```
//? Une fonction type querySelector pour récupérer UN élément (le 1er trouvé) par sa classe
let pSuper = document.querySelector('.super');
console.log('pSuper querySelector + class',pSuper);
```

### querySelectorAll()

Pour récupérer plusieurs éléments (par le nom de balise ou par leurs class css) dans une NodeList

```
//? Une fonction type querySelector pour récupérer TOUS les élément dans une NodeList
let allParagraphes = document.querySelectorAll('p');
console.log('allParagraphes querySelector + balise',allParagraphes);
let allSuper = document.querySelectorAll('.super');
console.log('allSuper querySelector + class',allSuper);
```

```
allSuper querySelector + class ▼NodeList(2) [h2.super, p.super] ⓘ
  ►0: h2.super
  ►1: p.super
  length: 2
  ►[[Prototype]]: NodeList
```

<https://github.com/jeff404/cours-js/tree/20-dom>

## Placer des éléments du DOM

Une fois que l'on arrive à sélectionner des éléments HTML (toute la page, le body, un titre, une div etc...) on va pouvoir utiliser des fonctions prévues pour gérer l'insertion / le placement de ces éléments dans la page.

### insertBefore( )

Cette fonction s'utilise sur un élément HTML sélectionné (généralement une <div>, un <form>, une <ul>, le <body>, etc...), prend en compte 2 éléments Html en paramètre pour placer l'un avant l'autre

```
///! Placer des elements dans une page web
//? Une fonction type querySelector pour récupérer UN (le 1er trouvé) élément par la
NodeList
let allParagraphs = document.querySelectorAll('p');
let laDiv = document.querySelector('.vide');
let premierH1 = document.querySelector('h1');
///! insertBefore, on selectionne 2 éléments pour placer l'un avant l'autre
document.body.insertBefore(allParagraphs[9], premierH1);
```

Page sans JS :

## D.O.M (Placer des éléments)

### Document Object Model

1 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

2 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

3 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

4 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

5 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

Page avec JS :

10 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

## D.O.M (Placer des éléments)

### Document Object Model

1 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

2 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

3 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

4 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## append( ) / appendChild( )

Généralement on va plutôt utiliser append ou appendChild pour placer un élément à la fin d'un autre (placer une div à la fin du body, placer un titre à la fin d'une div, etc...)

Pour l'exemple on va créer une div vide (donc elle ne s'affiche pas de base) avec du style

```
<div class="vide" style="background-color: midnightblue; color: beige;"></div>
```

```
laDiv.append(`Là c'est JS qui ajoute du texte dans la div`);  
// Append plutôt pensé pour ajouter du contenu à la volé au format string  
// si on a créé ou sélectionné un élément que l'on veut placer : ceci peut marcher  
laDiv.append(allParagraphs[4]);  
// Mais on a aussi la fonction appendChild;  
laDiv.appendChild(allParagraphs[0]);
```

Là c'est JS qui ajoute du texte dans la div

5 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

1 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

2 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

3 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

4 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores

## D.O.M (Placer des éléments)

### Document Object Model

#### Auteur :

Jean-François Pech

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date création :

03/03/2023

#### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## removeChild( )

Une fonction qui permet de retirer UN élément HTML du DOM

```
// //! On peut aussi supprimer un élément du DOM  
document.body.removeChild(allParagraphs[9]);
```

Ici on sur le body on supprime le 10è paragraphe

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

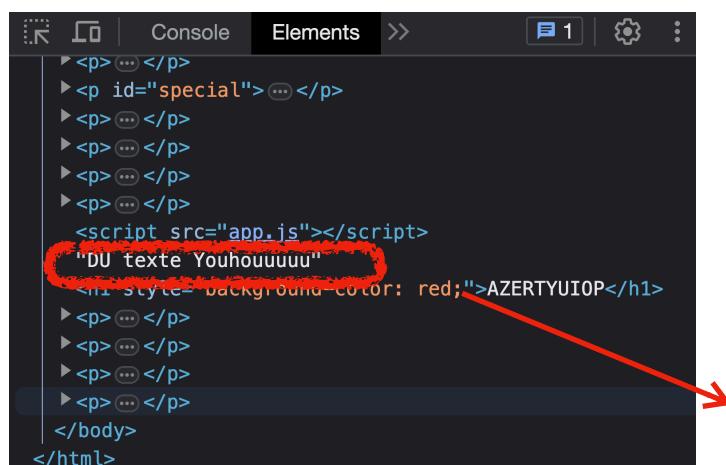
## Créer - Modifier - Supprimer des éléments du DOM

`createTextNode()`

Une fonction pour créer du texte brut dans une page web

```
//! Créer des éléments de texte
const newTxt = document.createTextNode('DU texte Youhouuuuu');
document.body.append(newTxt); //!Créer c'est bien mais il faut
placer
```

Dans l'inspecteur :



The screenshot shows the DOM structure of a page. A red circle highlights the text node 'DU texte Youhouuuuu'. A red arrow points from this node to the browser's preview pane on the right.

10 - Lorem ipsum dolor sit amet  
consectetur adipisicing elit.  
Explicabo deserunt magnam  
voluptas placeat, deleniti sed libero.  
Explicabo, officia dolores  
assumenda, exercitationem quas  
numquam eligendi, sit obcaecati  
ullam dignissimos quia suscipit?

DU texte Youhouuuuu

**Auteur :**  
Jean-François Pech  
**Relu, validé & visé par :**  
Jérôme CHRETIENNE  
Sophie POULAKOS  
Mathieu PARIS

**Date création :**

03/03/2023

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou  
rediffusion, totale ou partielle, de ce document  
ou de son contenu par quelque procédé que ce  
soit est interdite sans l'autorisation expresse,  
écrite et préalable de l'ADRAR.

## createElement( )

Cette fonction permet de créer n'importe quel élément HTML en précisant le nom de sa balise ( le tag Name)

Ici on crée un titre h1, on modifie quelques propriétés et on le place dans la page

```
//!Créer n'importe quel element HTML
const newH1 = document.createElement('h1');//phase 1 creation
newH1.innerText = "AZERTYUIOP";//phase2 remplissage
newH1.style.backgroundColor = 'red';
document.body.append(newH1);//phase 3 on place dans la page
```



AZERTYUIOP

⚠ Ne pas oublier de placer les éléments créés dans la page web 😊

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : Créer des éléments - Afficher un profil utilisateur

Nous avons cet objet userData et via JS on doit l'afficher dans la page web

```

///! EXO 20.1
//TODO: via JS afficher le profil utilisateur dans la page web
const userData = {
  name: 'John delavega',
  email: 'john.doe@example.com',
  age: 25,
  dob: '08/02/1989',
  active: true,
  img:'https://www.boredpanda.com/blog/wp-content/uploads/2022/06/funny-low-cost-
cosplay-pics-62a744d39c80a_700.jpg'
};

```



**John delavega**

john.doe@example.com

25

08/02/1989

true

## Solution Possible

```
// JS qui va customiser la div du profile utilisateur
let divUser = document.querySelector('.userProfile');
divUser.style.backgroundColor = `background-color: #4158D0`;
divUser.style.backgroundImage = `linear-gradient(43deg, #4158D0 0%, #C850C0 46%, #FFCC70 100%)`;
divUser.style.color = `white`;
divUser.style.width = '500px';
divUser.style.margin = 'auto';
divUser.style.padding = '2rem';
//JS crée une image, renseigne la src , modif taille
const imgTemplate = document.createElement('img');
imgTemplate.src = userData.img;
imgTemplate.style.height = '500px';
imgTemplate.style.width = '500px';
divUser.append(imgTemplate);
// JS crée le titre du name
const nameTemplate = document.createElement('h1');//phase 1 creation
nameTemplate.innerText = userData['name'];
divUser.append(nameTemplate);
// JS crée le titre du email
const emailTemplate = document.createElement('h2');//phase 1 creation
emailTemplate.innerText = userData.email;
divUser.append(emailTemplate);
// JS crée le titre du age
const ageTemplate = document.createElement('h2');//phase 1 creation
ageTemplate.innerText = userData.age;
divUser.append(ageTemplate);
// JS crée le titre du dob
const dobTemplate = document.createElement('h2');//phase 1 creation
dobTemplate.innerText = userData.dob;
divUser.append(dobTemplate);
// JS crée le titre du active
const activeTemplate = document.createElement('h2');//phase 1 creation
activeTemplate.innerText = userData.active;
divUser.append(activeTemplate);
```

⚠️ À optimiser avec une ou plusieurs fonctions ????????

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : Créer des éléments - fonction d'ajout de texte

```
///! EXO 20.1
//TODO: Créer une f° ajouterTexte qui prend 2 paramètres : pseudo et duTexte
//TODO: La fonction a pour but :
//TODO: de créer puis remplir et enfin placer un paragraphe contenant pseudo et
duTexte, dans la page
// TODO (Bonus) : Dans le paragraphe le pseudo est affiché en gras
```

exemple d'utilisation :

```
ajouterTexte('Daniel','Gracia');
ajouterTexte('Jarry','Borne');
ajouterTexte('JCVD','OK');
ajouterTexte('Dongue','Rodrigue');
```

Dans la page web :

**Daniel - Gracia**  
**Jarry - Borne**  
**JCVD - OK**  
**Dongue - Rodrigue**

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution possible

```
function ajouterTexte(unPseudo, duTexte){  
    const nouveauTexte = document.createElement('p');  
    nouveauTexte.innerHTML = `<strong>${unPseudo}</strong> - ${duTexte}`;  
    document.body.append(nouveauTexte);  
};
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Modifier les attributs des éléments du DOM

Accéder aux attributs

### Modifier attribut style

Comme pour les objets, une fois qu'on a sélectionné ou créé un élément HTML, on va pouvoir accéder à ses propriétés, certaines de ces propriétés vont nous permettre de modifier des attributs HTML.

Par exemple on peut accéder à l'attribut style des éléments et ensuite accéder à toutes les propriétés Css (⚠️ les propriétés Css ont une syntaxe en camelCase en JS)

```
let firstTitle = document.querySelector('h1');
console.log(firstTitle);

firstTitle.style.backgroundColor = 'blue';
firstTitle.style.color = 'beige';
```

## D.O.M (Modif Attributs)

Et si on vérifie via l'inspecteur du navigateur :

```
<h1 class="laClasse" style="background-color: blue; color: beige;">
D.O.M (Modif Attributs)</h1> == $0
```

## Modifier attribut class

Un autre attribut utile pour pouvoir gérer dynamiquement le style des éléments, c'est l'attribut `className` dans lequel on renseigne le nom d'une classe CSS que l'on veut ajouter à l'élément. Admettons que nous avons déjà préparé une classe CSS

```
.laClasse{  
    color: chartreuse;  
}
```

```
let firstTitle = document.querySelector('h1');  
console.log(firstTitle);  
  
firstTitle.className = 'laClasse';
```

## D.O.M (Modif Attributs)

Dans l'inspecteur du navigateur :

```
<h1 class="laClasse">D.O.M (Modif Attributs)</h1>
```

### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Fonction classList

Une autre option pour pouvoir gérer les classes c'est de passer par la propriété `classList`, qui donne accès à principalement trois fonction natives permettant d'ajouter, de supprimer ou de toggle des class CSS sur un élément HTML. (Il ya aussi une fonction `replace` pour remplacer des classes)

Bien sûr cela sous entend que nous avons déjà préparé les classes CSS.

## Modifier ou créer n'importe quel attribut

Avec la fonction `setAttribute()` qui s'utilise sur un élément HTML, on va pouvoir lui renseigner 2 paramètres, en premier quel attribut on veut modifier ou créer (si l'attribut existe déjà côté html ce dernier est modifier sinon il est créé), cette fonction permet aussi (dans des cas spécifiques) de créer les attributs de votre choix (beaucoup de framework ont des fonctionnalités basées sur des attribut spécifique côté template HTML).

Dans l'exemple ci-dessous on part du principe qu'on a une classe css nommée « democlass »

```
element.setAttribute("class", "democlass");
```

Ici exemple dans le cas où dans notre variable `element` (admettons nous avons sélectionné un lien `<a>`) on pourrait modifier son attribut `href`

```
element.setAttribute("href", "www.google.com");
```

Inspecter le D.O.M via le navigateur pour vérification

**Auteur :**  
Jean-François Pech  
**Relu, validé & visé par :**  
 Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date création :**

03/03/2023

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Gérer les évènements du DOM

Pour pouvoir prétendre à des application totalement « réactive » il va nous falloir cette dernière étapes à savoir, le fait de pouvoir réagir lorsque l'utilisateur déclenche un évènement dans la page. (Le click sur un bouton, le fait de scroller ou encore envoyer un formulaire).

La fonction addEventListener qui prend 2 arguments en paramètres, une chaîne de caractère pour mentionner quel évènement on doit surveiller (click, scroll, etc...) et en second paramètre, une fonction qui va réunir plusieurs instructions.

```
//? Mode f° => (anonyme + fléchée)
let leH1 = document.querySelector('#mainTitle');

leH1.addEventListener('click', ()=>{
    console.log('ok ca click');
});
```

Dans l'exemple ci-dessus on sélectionne un titre h1 via son id, ensuite sur cet élément HTML on écoute le click et en réaction on fait un affichage en console. (Ici on utilise une fonction fléchée en second paramètre)

```
//? Mode fonction anonyme classique
leH1.addEventListener('click',function(){
    console.log('ok ca click');
});
```

Ci-dessus avec une syntaxe classique de fonction mais également anonyme (généralement on utilise des fonctions anonymes dans les addEventListener, on n'a pas vraiment besoin de rappeler ces fonctions dans d'autres parties du programme), on peut aussi exécuter une fonction déclarée ailleurs dans le programme comme ci-dessous 

```
// ? la fonction est en dehors
function reactClick(){
    console.log('ok ca click');
}

leH1.addEventListener('click',reactClick());
```

### Auteur :

Jean-François Pech

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date création :

03/03/2023

### Date révision :

10/03/2023



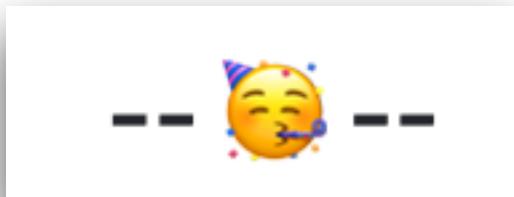
Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : réagir au click

Dans une page web mettre en place un titre h1, faire en sorte que lorsqu'on click sur le titre cela modifie son texte

# D.O.M Events

Une fois cliqué :

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution possible

```
let leH1 = document.querySelector('#mainTitle');
let selectTitle = false; //! Un booléen pour savoir si le titre a été cliqué ou non
leH1.addEventListener('click', ()=>{
    console.log('ok ca click');

    leH1.innerText = selectTitle ? '-- 😊 --':'D.O.M Events'; //? condition ternaire si
selectTitle est vrai alors smiley sinon dom Events
    selectTitle = !selectTitle; // ? à chaque click le booléen passe à son inverse
(pour faire le re click)
});
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023

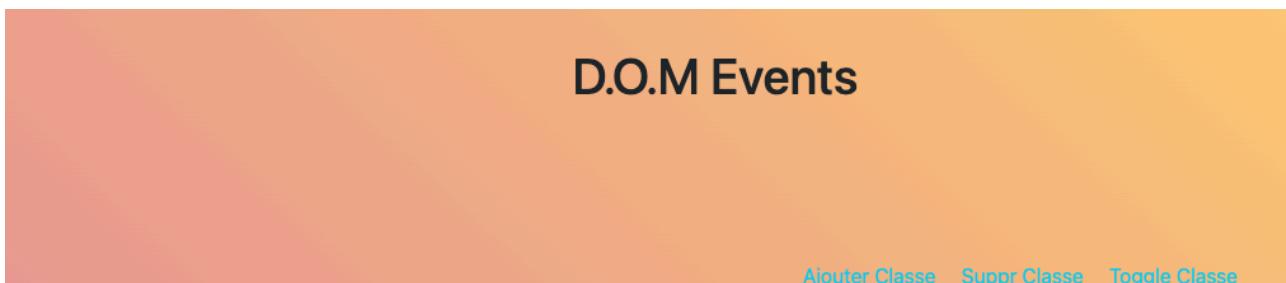


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice réagir au click 2

Dans une page web mettre en place 1 titre et trois liens ou boutons  
En css créer une classe

En utilisant les principales fonction de classList ,via JS, faire en sorte que le premier lien AJOUTE la classe css au titre de la page, le second lien SUPPRIME la classe et le troisième lien fais un TOGGLE de la classe sur le titre.



Avec la classe appliquée

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution possible

La classe en Css :

```
.laClasse {  
    color: chartreuse;  
    font-style: oblique;  
}
```

En JS, On réagit au click et on utilise classList et ses fonctions pour appliquer ou non la classe Css nommée « laClasse »

```
const leTitre = document.querySelector("h1");  
const lesLiens = document.querySelectorAll("a");  
lesLiens[0].addEventListener("click", function(){  
    leTitre.classList.add("laClasse");  
    document.body.classList.add("bodyBg");  
});  
lesLiens[1].addEventListener("click", function(){  
    leTitre.classList.remove("laClasse");  
});  
lesLiens[2].addEventListener("click", function(){  
    leTitre.classList.toggle("laClasse");  
});
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023

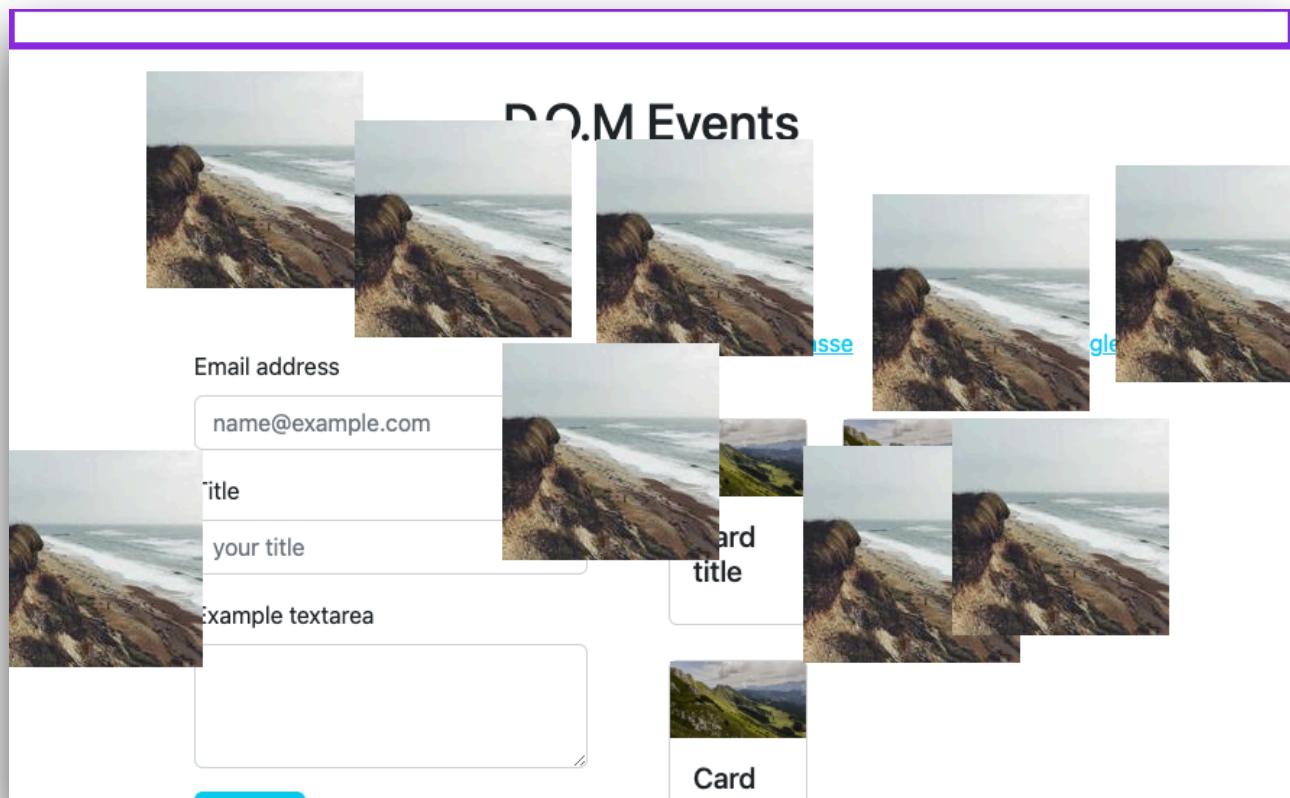


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exo : réagir au click 3 (capter l'event)

Dans une page web faire en sorte de récupérer les coordonnées X et Y du click et créer une image à cet endroit là.

Pensez à mettre un paramètre dans la fonction du addEventListener afin de l'afficher en console (pour capter l'évènement)

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution Possible

```
document.addEventListener('click', function(eventClick){  
    console.log(eventClick);  
    console.log('Les coord : ',eventClick.x, eventClick.y);  
    // On Crée une image  
    const monImg = document.createElement('img');  
    const tailleImg = 150;  
    // On rajoute une src à l'image  
    monImg.setAttribute('src', `https://picsum.photos/${tailleImg}/${tailleImg}`);  
    // On modifie le type de position de l'image  
    monImg.style.position = 'absolute';  
    // On modifie la position top de l'image (la division par 2 pour que l'image se  
    crée centrée)  
    monImg.style.top = eventClick.y - tailleImg /2 +'px';  
    // On modifie la position left de l'image  
    monImg.style.left = eventClick.x -tailleImg /2 +'px';  
    //On place l'image fraîchement créée dans le body de la page  
    document.body.append(monImg);  
});
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : réagir au focus et au Blur

Dans une page web le focus correspond à ce que l'utilisateur sélectionne (faites le test de faire une recherche google et d'appuyer plusieurs fois sur la touche Tab du clavier) ou alors plus classique dans la plupart des navigateur quand on écrit dans des champs de texte ces derniers sont entourés en bleu

Avec focus

Example textarea

Sans (donc l'évènement Blur, l'inverse de focus l'utilisateur ne sélectionne plus cet élément html)

Example textarea

Dans une page web rajouter un input de type text, sur cet input quand on réagit au focus, on modifie son background Color et la couleur du texte,

On doit aussi réagir au blur pour remettre une couleur de background transparent et la couleur du texte en noir.

Title

azertyuiop

Title

azertyuiop

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution possible

```
const formTitleInput = document.querySelector('#titleInput');
// TEST avec FOCUS
formTitleInput.addEventListener("focus", function() {
    formTitleInput.style.backgroundColor = "royalBlue";
    formTitleInput.style.color = "white";
});
// TEST avec Blur
formTitleInput.addEventListener("blur", function() {
    formTitleInput.style.backgroundColor = "transparent";
    formTitleInput.style.color = "black";
});
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

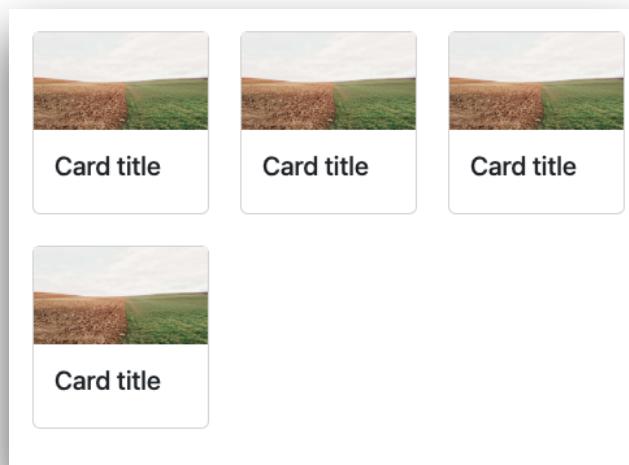
10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : réagir au Load

Dans une page web rajouter plusieurs images (de haute qualité 😊)



Sélectionner **TOUTES** les images dans une variable

ensuite utiliser **Array.from( laVariableAtransformer )** pour transformer cette HTMLCollection ou NodeList en **Array**, sur cet array utiliser **map( )**, pour placer un addEventListener qui réagit au « load » et qui affiche un message en console

Vous pouvez utiliser l'index de la fonction map( ) pour numérotter les images

En console :

```
Image numéro : 3 - vient de finir de charger. app.js:131
Image numéro : 0 - vient de finir de charger. app.js:131
Image numéro : 2 - vient de finir de charger. app.js:131
Image numéro : 1 - vient de finir de charger. app.js:131
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution Possible

```
let lesImages = document.querySelectorAll('img');
console.log(lesImages);
let tabImg = Array.from(lesImages);
console.log(tabImg);
tabImg.map(function (uneImage, index) {
    uneImage.addEventListener("load", function () {
        console.log(`Image numéro : ${index} – vient de finir de charger.`);
    });
});
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : réagir à mouseleave

Lorsque la souris de l'utilisateur s'en va d'un élément html.

Dans une page web rajouter un titre (h1, h2 ou h3 au choix) il est en display none en CSS.

Ensuite sur toute la page surveiller l'évènement mouseleave de manière à faire apparaître le titre en mettant son display en « block » (ajouter d'autre modifications du titre via js notamment au niveau du style)

Dès que le curseur de la souris quitte la page :



**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution Possible

```
const h3Title = document.querySelector('#mouseOut');
// const mainContainer = document.querySelector('#mainContainer');
// let lesImagesFirst = document.querySelectorAll(`img`)[0];

console.log(h3Title);
document.addEventListener('mouseleave', ()=>{
    //? Comme vu pour les objets on peut accéder aux propriétés dans l'objet style de l'element
    h3Title.style.display = 'block';
    h3Title.style.color = 'red';
    h3Title.style.backgroundColor = 'chartreuse';
    h3Title.innerText = 'Tu as gagné 1 millions de Dollars';
    h3Title.style.textAlignment = 'center';
});
```

Ici on place l'écouteur d'évènements sur document donc toute la page web mais on peut bien entendu le faire sur n'importe quel élément sélectionné.  
(Il existe aussi l'évènement inverse avec mouseenter)

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : réagir au scroll

Sur toute la page écouter l'évènement scroll, faire un console log de l'évènement ainsi que des console log pour ces variables

Mettez en place une variable scrollMax, dans laquelle on soustrait à document.body.scrollHeight, la variable innerHeight,

Faire une variable onEstOu dans laquelle on stock un pourcentage à partir de scrollY et scrollMax

Puis faire en sorte d'assigner la variable onEstOu à la largeur de la div qui a la class « bar »

```
console.log(`  
    Hauteur page : ${document.body.scrollHeight}  
    Hauteur affichage : ${innerHeight}  
    Scroll Position : ${scrollY}`);  
});
```



Et en console :

```
▶ Event  
  
Hauteur page : 5000  
Hauteur affichage : 995  
Scroll Position : 1537  
pourcentage de scroll :38.37702871410736 %
```

## Solution possible

```
const laBar = document.querySelector(".bar");
document.addEventListener("scroll", function (event) {
    console.log(event);
    //Le scrollMax = hauteur de la page - hauteur de affichage
    const scrollMax = document.body.scrollHeight - innerHeight;
    // On fait un pourcentage du scroll de l'utilisateur
    const onEstOu = (scrollY / scrollMax) * 100;
    //Enfin on assigne ce pourcentage de scroll
    //à la width(%) du style de la bar.
    laBar.style.width = onEstOu + "%";
    console.log(`\n      Hauteur page : ${document.body.scrollHeight}\n      Hauteur affichage : ${innerHeight}\n      Scroll Position : ${scrollY}\n      pourcentage de scroll : ${onEstOu} %`);
});
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023

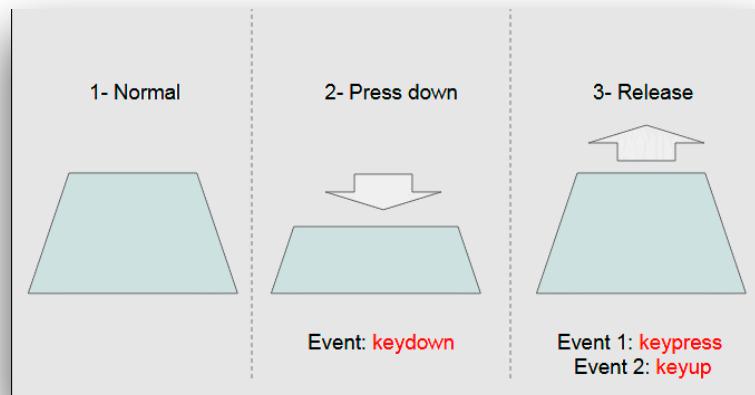


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : réagir au clavier

Dans ce cas là plusieurs évènements s'offrent à nous, keyup, keypress ou keydown, cela correspond au différentes phase lorsque l'on presse une touche du clavier.

Plus explicite :



Généralement pour pouvoir exploiter ce que tape l'utilisateur on va privilégier l'évènement keyup.  
 (Keydown rarement (les long jump dans sonic 2 🎮 ?)

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Dans une page web : rajouter un formulaire (au moins un textarea)

Dans votre programme JS :

No Spoil : )

## D.O.M Events

Email address [Ajouter Classe](#) [Suppr Classe](#) [Toggle Classe](#)

S'AFFICHE ICI ➡ azertyuiop

Title

your title

CE QU'ON ECRIT ICI 📸

Example textarea

azertyuiop

Submit

Card title

Card title

Card title

Card title

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution Possible

```
const monTxt = document.querySelector("textarea");
const rendu = document.querySelector("#formRender");

monTxt.addEventListener("keyup", function() {
    rendu.innerHTML = monTxt.value;
});
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : Réagir au clavier 2

Dans une page web on a un input ou textarea, et un bouton, faire en sorte que :

Au bout de 5 lettres tapées au clavier cela Désactive le bouton :

Example textarea

rrrrr

**Submit**

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Web Storage

Dans le navigateur on peut profiter de plusieurs espaces pour pouvoir sauvegarder des données, vous connaissez déjà (surtout depuis RGPD) les cookies qui permettent par exemple d'enregistrer un email et mot de passe dans un formulaire, ou reprendre l'épisode d'une série au moment où nous l'avions quitté, etc ...

Ce schéma récapitule les différentes caractéristiques de ces principaux espaces de stockage.





Criteria	Local Storage	Session Storage	Cookies
Storage Capacity	5-10 mb	5-10 mb	4 kb
Auto Expiry	No	Yes	Yes
Server Side Accessibility	No	No	Yes
Data Transfer HTTP Request	No	No	Yes
Data Persistence	Till manually deleted	Till browser tab is closed	As per expiry TTL set

Dans l'exemple suivant nous allons nous intéresser au local Storage (la façon de l'utiliser est assez similaire pour la session et les cookies)

**Auteur :**  
Jean-François Pech  
**Relu, validé & visé par :**  
 Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date création :**

03/03/2023

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

⚠ Gardez en tête que ces espaces de stockage ne sont pas définitif, l'utilisateur peut toujours aller dans les paramètres du navigateur (ou si il est navigation privée) et supprimer ces espaces de stockage, de plus ils ont une capacité de stockage limitée.

Local Storage, évidemment il y a un objet localStorage avec des fonctions natives de JS permettant de gérer cet espace de mémoire commun à tous les navigateurs.

### La fonction setItem( )

qui prend 2 paramètres, (une clé et une valeur), ci dessous dans le local Storage on sauvegarde une clé 'lastname' avec une valeur à 'smith'

```
localStorage.setItem("lastname","Smith");
```

### La fonction getItem( )

qui prend 2 paramètres, (une clé), ci dessous dans le local Storage on récupère le contenu de la clé 'lastname'

```
localStorage.getItem("lastname");
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023

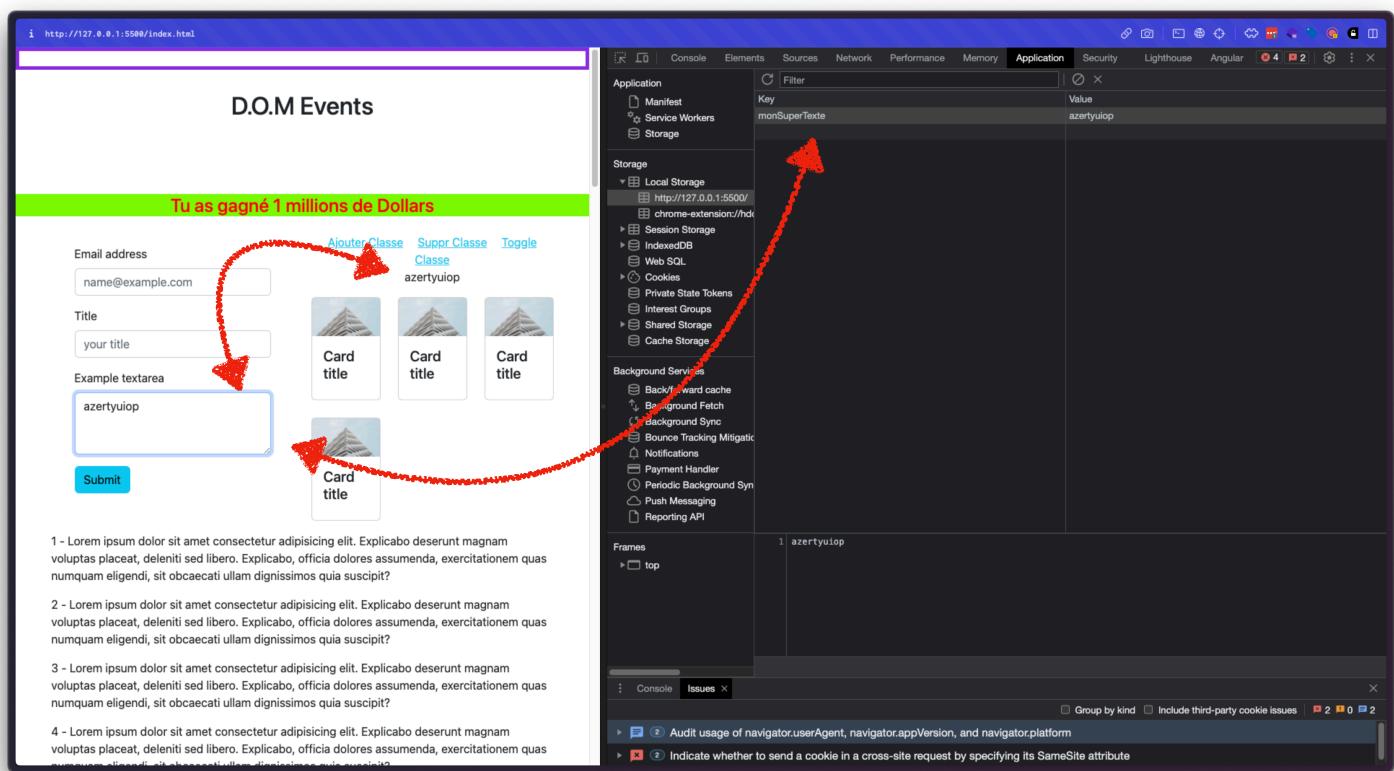


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : LocalStorage

En se basant sur l'exercice précédent (le mini éditeur de texte)

Dans une page web avec un formulaire (ou au moins un textarea)



The screenshot shows a browser window with the URL <http://127.0.0.1:5500/index.html>. The page title is "D.O.M Events". The content includes a success message "Tu as gagné 1 millions de Dollars", an email input field ("name@example.com"), a title input field ("your title"), and a textarea ("Example textarea") containing "azertyuiop". Below the textarea are three cards labeled "Card title". On the right, the browser's developer tools are open, specifically the Application tab under Storage. A red arrow points from the "monSuperTexte" key in the storage list to its value "azertyuiop" in the details panel. Another red arrow points from this value back to the "azertyuiop" text in the textarea.

### Dans votre programme JS :

- Stocker l'input ou textarea dans une variable monTxt
- Stocker la div (ou le titre ou paragraphe etc..) dans une variable rendu
- À la valeur du textarea assigner ce qu'on récupère dans le local Storage sous le nom (la clé) « monSuperTexte »
- Ensuite mettre en place une condition : SI la valeur de monTxt est définie ALORS on le innerText du rendu récupère dans le local Storage sous le nom (la clé) « monSuperTexte »
- Après cette condition mettre en place une écoute de l'événement du clavier sur monTxt et faire :
  - Placer dans le local Storage la valeur de monTxt sous le nom « monSuperTexte »
  - Afficher la valeur de monTxt dans le rendu

En console dans l'onglet Application :

<b>Auteur :</b>	<b>Date création :</b>
Jean-François Pech	03/03/2023
<b>Relu, validé &amp; visé par :</b>	<b>Date révision :</b>
Jérôme CHRETIENNE Sophie POULAKOS Mathieu PARIS	10/03/2023




Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

The screenshot shows the Chrome DevTools Application tab open. On the left, there's a sidebar with sections for Application (Manifest, Service Workers, Storage), Storage (Local Storage, Session Storage), and Network (Request, Response). The Local Storage section is expanded, showing two items: 'http://127.0.0.1:5500/' and 'chrome-extension://hdcd'. The main area displays a table with columns for Key and Value. One entry is visible: 'monSuperTexte' with the value 'lkjfhskgjhsfkgjhksfjdh'. There are also buttons for Filter, Clear, and Close.

Key	Value
monSuperTexte	lkjfhskgjhsfkgjhksfjdh

Quand on recharge la page ou quitter et relancer le navigateur le mini éditeur de texte est pré-rempli et son affichage aussi

Jean-François Pech

03/03/2023

100-100-100-100



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Solution Possible

```
const monTxt = document.querySelector("#formMessage");
const rendu = document.querySelector("#formRender");
// On va pré remplir le textarea avec ce que l'on récupère dans le localStorage
monTxt.value = localStorage.getItem('monSuperTexte');
//Si monTxt.value est définit alors on remplit la Div avec ce qu'on récupère dans le
localStorage
if(monTxt.value){
    rendu.innerText = localStorage.getItem('monSuperTexte');
}
//On détecte ce que tape l'utilisateur dans le textarea
monTxt.addEventListener("keyup", function() {
//On enregistre ce que tape l'utilisateur dans le localStorage sous le nom
"monSuperTexte"
localStorage.setItem('monSuperTexte',monTxt.value);
//On affiche ce que tape l'utilisateur dans la div
    rendu.innerHTML =monTxt.value;
});
```

**Auteur :**

Jean-François Pech

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date création :**

03/03/2023

**Date révision :**

10/03/2023

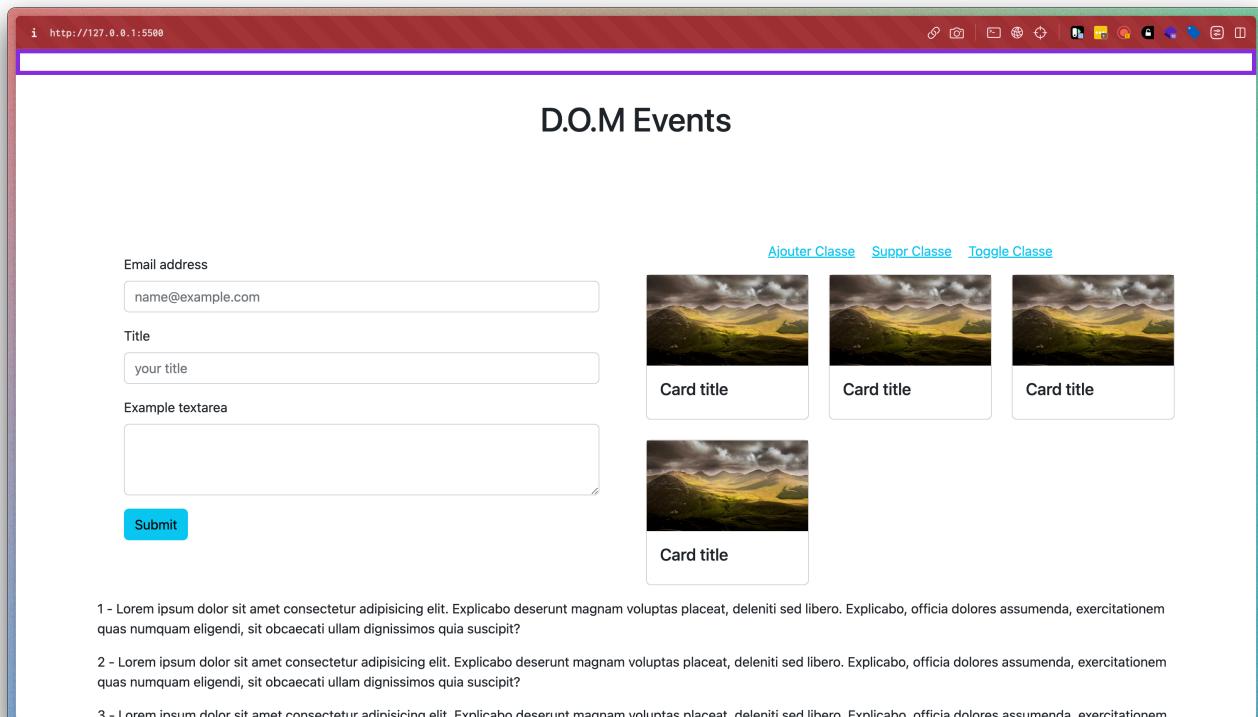


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice DOM Settimeout

Dans une page web, en utilisant la fonction settimeout, faire en sorte qu'au bout de 3 secondes (soit 3000 ms), un titre apparaît sur la page et des modifications du style sont effectuées.

La page quand elle est chargée :



The screenshot shows a web page with the URL <http://127.0.0.1:5500>. The page title is "D.O.M Events". On the left, there is a form with fields for "Email address" (containing "name@example.com"), "Title" (containing "your title"), and "Example textarea". Below the form is a "Submit" button. On the right, there are four cards, each featuring a placeholder image of a landscape and the text "Card title". Above the cards is a row of buttons: "Ajouter Classe", "Suppr Classe", and "Toggle Classe". Below the cards is another card with the same layout. At the bottom of the page, there are three paragraphs of lorem ipsum text.

1 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

2 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?

3 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

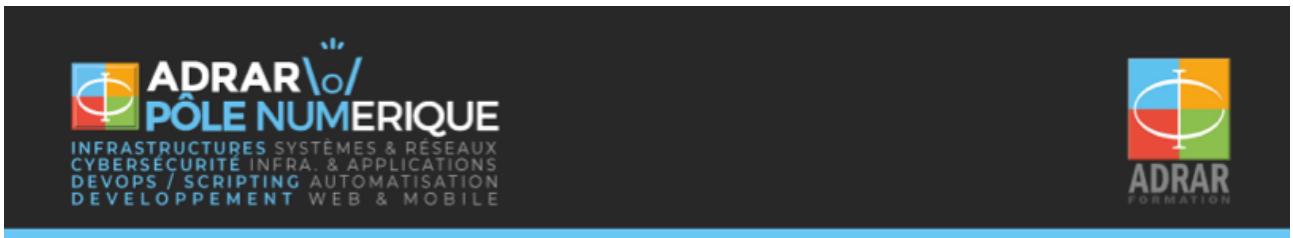
Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.



La page au bout de 3 secondes :

A screenshot of a web application window titled "D.O.M Events". The title bar shows the URL "http://127.0.0.1:5500". The main content area has a gradient background from pink to orange. At the top, it says "Welcome 2 the DOM". Below this is a form with fields for "Email address" (containing "name@example.com"), "Title" (containing "your title"), and "Example textarea". A "Submit" button is at the bottom of the form. To the right of the form is a grid of four cards, each featuring a landscape image and the text "Card title". Above the grid are three buttons: "Ajouter Classe", "Suppr Classe", and "Toggle Classe". At the bottom of the page, there are three paragraphs of placeholder text: "1 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?", "2 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?", and "3 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo deserunt magnam voluptas placeat, deleniti sed libero. Explicabo, officia dolores assumenda, exercitationem quas numquam eligendi, sit obcaecati ullam dignissimos quia suscipit?".

## B.O.M

Exemple géoloc ?

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## JS & Firebase

Nous allons utiliser Javascript avec une base de donnée automatiquement hébergée en ligne, Firebase, il s'agit d'une solution proposée par google contenu plein d'outils dont des systèmes de base de données, au même titre que AWS (Amazon Web Service) Microsoft Azure, MongoDB ou encore le concurrent direct et open-source, Supabase.

Il s'agit d'un BAAS (Backend As A Service), c'est à dire que plutôt que d'avoir à créer et gérer nous même la partie Backend de notre application nous utiliserons celui que Google (ou autre) nous met à disposition.

Dans Firebase, il y a donc plusieurs outils mais pour l'exemple nous allons utiliser la REALTIME DATABASE (une base de données en temps réel) qui est en NoSQL (Not Only SQL), en l'occurrence nous allons travailler sur une BDD organisée en JSON.

Plus d'infos ici pour comparer avec MySQL :

<https://blog.back4app.com/fr/firebase-database-contre-mysql/>

	Firebase	MySQL
<b>Brève description</b>	Plateforme de développement d'applications de Google.	Base de données SQL open-source
<b>Prix</b>	Démarrage gratuit Payez au fur et à mesure par la suite	Téléchargement gratuit
<b>Open-Source</b>	Non	Oui
<b>Verrouillage du fournisseur</b>	Oui	Non
<b>Service géré</b>	Oui	Hébergement géré disponible sur Oracle, AWS, Digital Ocean, etc.
<b>Traitement des données</b>	Firebase traite efficacement les grands ensembles de données. Il utilise des magasins à colonnes larges, à valeurs clés, à documents ou à graphes et dispose de schémas dynamiques pour faciliter les données non structurées.	MySQL est basé sur des tables et possède des schémas prédéfinis. C'est un choix privilégié pour le traitement de données complexes.
<b>Architecture</b>	Firebase est une base de données NoSQL qui synchronise et stocke les données en temps réel.	MySQL est un système de gestion de base de données relationnelle open source qui repose sur SQL, le langage spécifique au domaine.
<b>Assistance linguistique</b>	Beaucoup moins que MySQL.	Les langages de programmation pris en charge par MySQL sont bien plus nombreux que ceux pris en charge par Firebase. Ces langages comprennent Python, C++, Ada, etc.
<b>Évolutivité</b>	Firebase évolue horizontalement.	MySQL évolue verticalement.

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

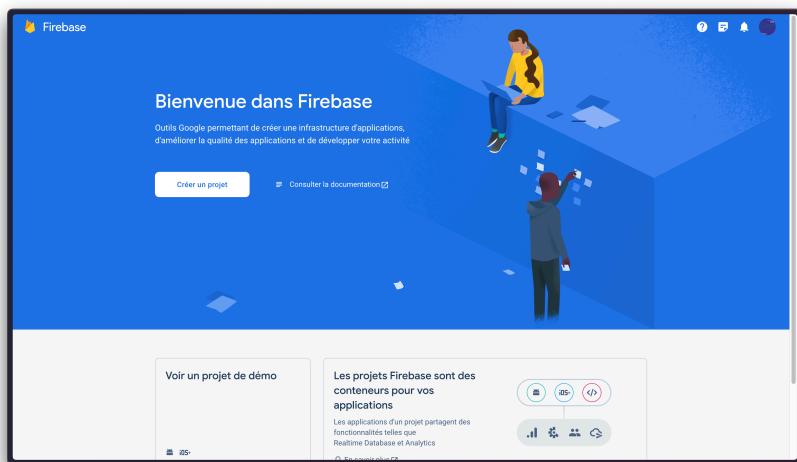
10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Set-up de firebase

Se rendre sur le site de la console Firebase  
<https://console.firebaseio.google.com>



Ensuite on crée un projet dans Firebase :



### Auteur :

Jean-François Pech

### Date création :

03/03/2023

### Relu, validé & visé par :

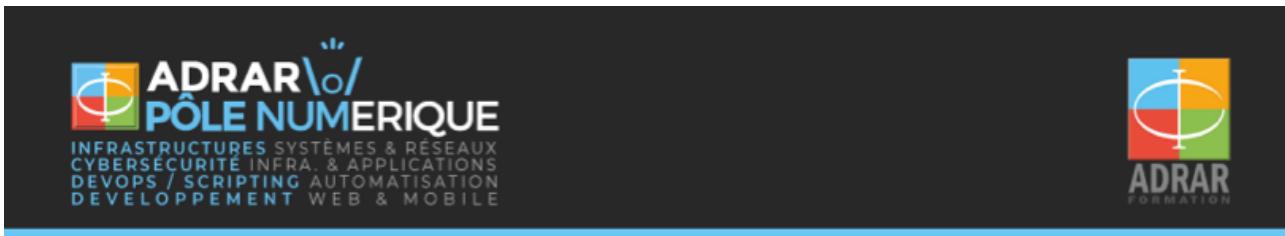
Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date révision :

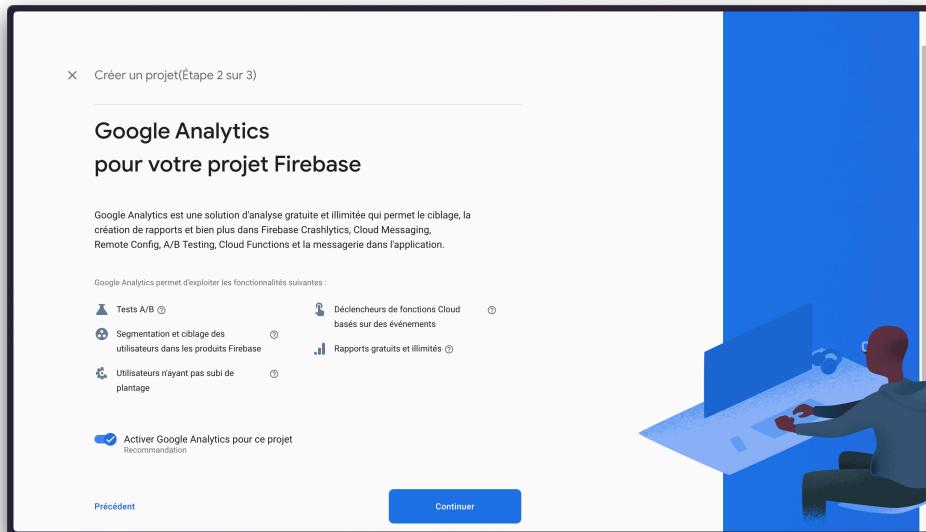
10/03/2023



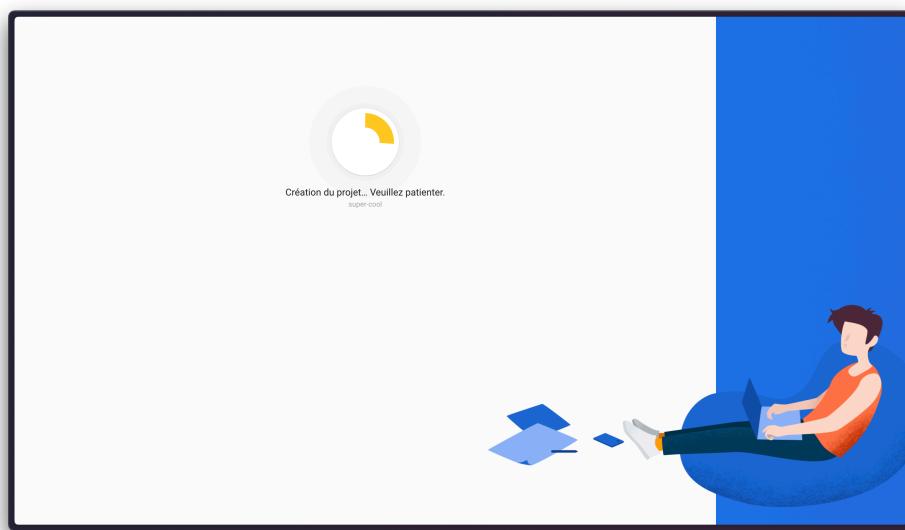
Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

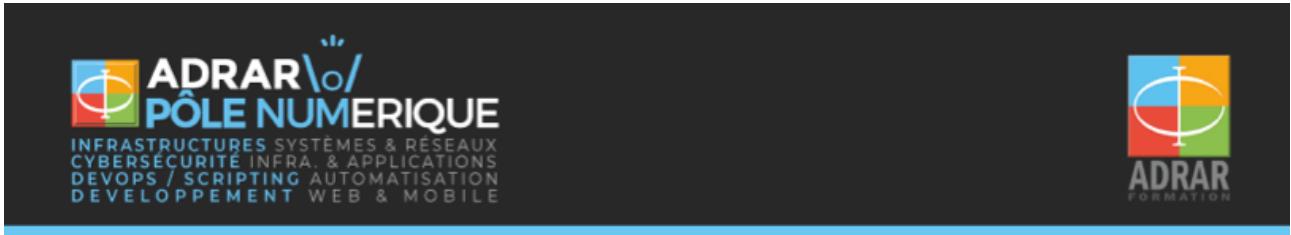


Optionnel : on peut activer Google Analytics, pour ce projet test on peut le désactiver



Ensuite Firebase crée le projet





On atterri alors sur la page d'accueil de notre projet ici « super-cool »

A screenshot of the Firebase console interface. On the left, a sidebar shows navigation options like "Créer", "Publier et surveiller", "Analytics", and "Engager". The main area displays a landing page for the "super-cool" project, which is a "Formule Spark". The page features a large blue banner with the text "Lancez-vous en ajoutant Firebase à votre application" and icons for iOS+, Android, and web development. Below the banner, there's a callout about real-time synchronization and two small images related to user authentication and database queries.

Dans l'onglet à gauche on va créer une nouvelle base de données de type **RealTime Database**

A screenshot of the Firebase console interface. The left sidebar is open, showing the "Créer" section with various services: Authentication, App Check, Firestore Database, Realtime Database (which is highlighted), Extensions, Storage, Hosting, Functions, Machine Learning, and Remote Config. The main area shows a landing page for the "super-cool" project, similar to the one above, but with the "Realtime Database" service selected.

A footer section containing several pieces of information: "Auteur:" followed by "Jean-François Pech"; "Date création:" followed by "03/03/2023"; "Relu, validé &amp; visé par:" followed by three names with checkboxes; "Date révision:" followed by "10/03/2023"; and a copyright notice from "ADRAR PÔLE NUMÉRIQUE TECH' CARE" stating that reproduction is prohibited without permission. The footer also includes the ADRAR Formation logo.



## Créer une RealTime Database :

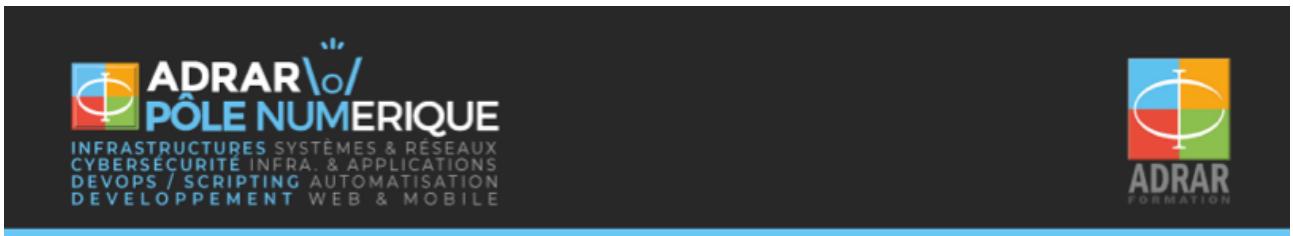
The screenshot shows the Firebase Realtime Database setup process. It starts with the main Firebase dashboard, then moves to the 'Realtime Database' configuration screen. On this screen, the location 'Etats-Unis (us-central1)' is selected. Below the configuration, there is a note about security rules in 'mode test'.

Ensuite choisir « démarrer en mode test »

The screenshot shows the 'Configure a database' step, specifically the 'Security Rules' section. The 'Mode test' option is selected. A note states: 'Par défaut, les règles de sécurité en mode test autorisent tout utilisateur disposant de la référence de votre base de données à afficher, modifier et supprimer toutes les données qu'elle contient pendant les 30 prochains jours'.

<b>Auteur :</b> Jean-François Pech	<b>Date création :</b> 03/03/2023	<b>Relu, validé &amp; visé par :</b> Jérôme CHRETIENNE Sophie POULAKOS Mathieu PARIS
<b>Date révision :</b> 10/03/2023		 ADRAR FORMATION

Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.



Nous arrivons sur notre base de donnée  
(Firebase nous a créée la base de donnée et elle est hébergée donc accessible par l'url (on pourrait aussi contacter la base via les api))

Firebase

Vue d'ensemble du p... |

Raccourcis de projet

Realtime Database

Catégories de produits

Créer

Publier et surveiller

Analytics

Engager

Tous les produits

Spark Sans frais 0 \$/mois Mettre à niveau

super-cool ▾

Realtime Database

Données Règles Sauvegardes Utilisation NOUVEAU

Protégez vos ressources Realtime Database des utilisations abusives telles que la fraude à la facturation et le hameçonnage Configurer App Check X

https://super-cool-3d83b-default.firebaseio.com/.json

Emplacement de la base de données : États-Unis (us-central1)

Ensuite on va se rendre dans les paramètres du projet afin d'enregistrer une application pour cette RealTime Database

Firebase

Vue d'ensemble du p... |

Raccourcis de projet

Realtime Database

Catégories de produits

super-cool ▾

Paramètres du projet

Utilisateurs et autorisations

Utilisation et facturation

Protégez vos ressources Realtime Database

**Auteur :**  
Jean-François Pech

**Date création :**  
03/03/2023

**Relu, validé & visé par :**  
Jérôme CHRETIENNE  
Sophie POULAKOS  
Mathieu PARIS

**Date révision :**  
10/03/2023

Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Dans les paramètres généraux, en scrollant tout en bas :  
 On va ajouter une application web donc l'icône « < / > »

Vos applications

Votre projet ne comporte aucune application

Pour démarrer, sélectionnez une plate-forme



On renseigne un nom pour notre application web (pas besoin de firebase Hosting ça sera peut être pour plus tard si on veut déployer l'application chez Firebase):

Ajouter Firebase à votre application Web

1 Enregistrer l'application

Pseudo de l'application

Configurez également **Firebase Hosting** pour cette application. [En savoir plus](#)

Hosting peut également être configuré plus tard, et vous pouvez commencer à l'utiliser sans frais à tout moment.

**Enregistrer l'application**

2 Ajouter le SDK Firebase



Ensuite Firebase nous fournit des codes d'accès pour pouvoir contacter notre RealTime Database, on va au moins récupérer la variable firebaseConfig qu'on placera dans notre programme JS

```
// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AI[REDACTED]W",
  authDomain: "[REDACTED].com",
  databaseURL: "https://[REDACTED].firebaseapp.com",
  projectId: "s[REDACTED]",
  storageBucket: "s[REDACTED].appspot.com",
  messagingSenderId: "1[REDACTED]",
  appId: "1[REDACTED]"
};
```

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Setup-JS

```
//La variable de config pour firebase
const firebaseConfig = {
  apiKey: "-----VOS INFORMATIONS FIREBASE-----",
  authDomain: "-----VOS INFORMATIONS FIREBASE-----",
  databaseURL: "-----VOS INFORMATIONS FIREBASE-----",
  projectId: "-----VOS INFORMATIONS FIREBASE-----",
  storageBucket: "-----VOS INFORMATIONS FIREBASE-----",
  messagingSenderId: "-----VOS INFORMATIONS FIREBASE-----",
  appId: "-----VOS INFORMATIONS FIREBASE-----",
};

firebase.initializeApp(firebaseConfig);
//On va créer une référence à notre BDD
const dbRef = firebase.database().ref();
// On va également faire une ref directement dans le noeud / "table" users
const usersRef = dbRef.child("users");

const addUserBtnUI = document.getElementById("add-user-btn");
addUserBtnUI.addEventListener("click", addUserBtnClicked);

const formUserUI = document.getElementById("add-user-form");
formUserUI.addEventListener("submit", (event) => event.preventDefault());

const formUserEditUI = document.getElementById("edit-user-module");
formUserEditUI.addEventListener("submit", (event) => event.preventDefault());

const userListUI = document.getElementById("user-list");
const userDetailUI = document.getElementById("user-detail");

readUserData();

function addUserBtnClicked() {

};

function readUserData() {

};

function userClicked(event) {

};

function editButtonClicked(event) {

};

function saveUserBtnClicked() {

};

function deleteButtonClicked(event) {

}
```

**Auteur :**

Jean-François Pech

**Relu, validé & visé par :**

- Jérôme CHRETIENNE
- Sophie POULAKOS
- Mathieu PARIS

**Date création :**

03/03/2023

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Setup-HTML

On va utiliser Firebase en mode CDN (pas besoin d'installation), ajouter les scripts de firebase à la fin du body de la page :

- firebase-app.js
- firebase-database.js

```
<!-- firebase sdk link goes here -->
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.4.2.firebaseio-app.js"></script>
<!-- <script src="https://www.gstatic.com/firebasejs/8.4.2/firebase-analytics.js"></script> -->
<script src="https://www.gstatic.com/firebasejs/8.4.2.firebaseio-database.js"></script>
<!-- JS de Bootstrap -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
       integrity="sha384-kenU1KFdBIe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4"
       crossorigin="anonymous"></script>
<script src="app.js"></script>
</body>
```

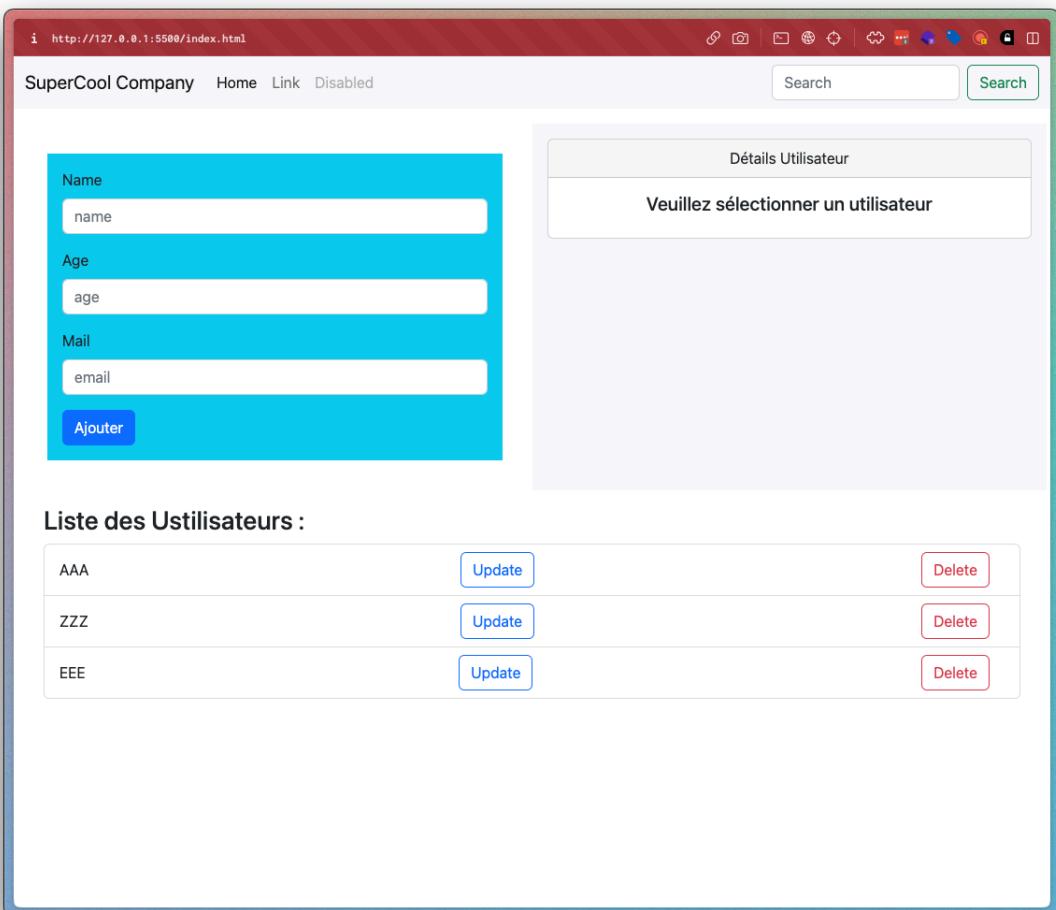
### Il nous faudra :

- 1 formulaire avec id « add-user-form » (le formulaire pour ajouter un user) :
  - 3 inputs avec la classe « user-input »
  - Ces 3. Inputs ont un attribut data-key chacun avec comme valeurs respectives (name, age, mail)
  - 1 button avec id « add-user-btn »
- 1 formulaire avec id « edit-user-module » (le formulaire pour modifier un user) :
  - 3 inputs avec la classe « edit-user-input »
  - Ces 3. Inputs ont un attribut data-key chacun avec comme valeurs respectives (name, age, mail)
  - 1 button avec id « edit-user-btn »
- 1 <div> avec id « user-detail » (là où on affiche les infos détaillées d'un user) :
- 1 <li> avec id « user-liste » (là où on affiche la liste de tous les users) :

## Setup-CSS

Le formulaire pour éditer un utilisateur sera masqué de base, JS se chargera de l'afficher ou non

```
/*Edit*/
#edit-user-module {
    display: none;
}
```

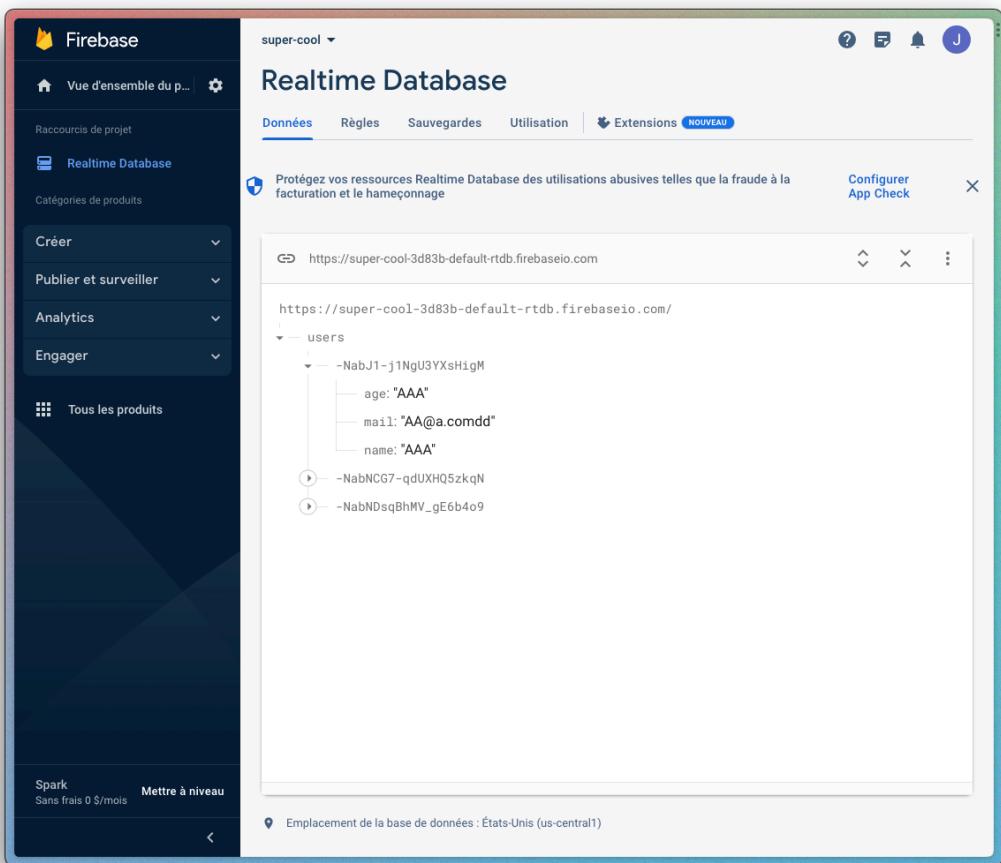


The screenshot shows a web application interface for managing users. On the left, there is a form for adding a new user with fields for Name, Age, and Mail, and a blue "Ajouter" button. On the right, there is a modal window titled "Détails Utilisateur" with the message "Veuillez sélectionner un utilisateur". Below this, there is a section titled "Liste des Utilisateurs:" displaying a table with three rows: AAA, ZZZ, and EEE. Each row has an "Update" button and a "Delete" button.

## Exercice : CREATE : Ajouter un nouvel utilisateur

```
//TODO 5: Dans la f° addUserBtnClicked, Récupérer TOUS LES INPUTS avec la classe user-input 1 variable addUserInputsUI (getElementsByClassName)
//TODO 6: Dans la f° addUserBtnClicked, créer une variable newUser (qui est un objet vide)
//TODO 7: Dans la f° addUserBtnClicked, faire une boucle for pour parcourir les input dans addUserInputsUI
//TODO 8: Dans la Boucle, Pour chaque éléments parcourus on récupère Dans 1 variable key = addUserInputsUI[i].getAttribute('data-key')
//TODO 9: Dans la boucle, 1 variable value = addUserInputsUI[i].value
//TODO 10: Dans la boucle, Pour chaque clé (âge, name, email) on l'associe à notre nouvel utilisateur : newUser[key] = value
//TODO 11: après le parcours des inputs, sur usersRef on va faire un push de newUser
//TODO 12: Dans la f° addUserBtnClicked, on console log un msg type nouvel utilisateur enregistré
//TODO 13: Dans la f° addUserBtnClicked, On console log le nom et l'âge du nouvel utilisateur
//TODO 14: Dans la f° addUserBtnClicked, On ré initialise le formulaire avec l'id add-user-form
```

Après avoir testé le formulaire d'ajout peut se rendre dans la console Firebase du projet pour constater que Friable a crée un noeud « users » avec un premier objet pour notre premier utilisateur avec un id généré automatiquement par Firebase :



The screenshot shows the Firebase Realtime Database interface. On the left, there's a sidebar with project settings and a 'Realtime Database' section. The main area is titled 'Realtime Database' and shows a single node under 'Données'. The node path is 'https://super-cool-3d83b-default.firebaseio.com/users/-NabJ1-j1NgU3YXsHigM'. Inside this node, there are three child nodes: 'age: "AAA"', 'mail: "AA@a.comdd"', and 'name: "AAA"'. The database is configured to use 'App Check'.

## Exercice : READ (Lecture de la BDD : Tous Les Utilisateurs)

```
/* LIRE TOUT LES USERS
//TODO : Dans la f° readUserData
//TODO : sur la variable usersRef on va utiliser une fonction .on()
//? Pour info .on() va s'utiliser comme un addEventListener
//TODO : 1er param de .on(), une string "value" (en gros dans la bdd on surveille si ya des changements de value)
//TODO : 2e param de .on(), une f° fléchée qui prend un paramètre snap
//? usersRef.on("value", (snap) => {});
//TODO : Dans la fonction fléchée : on va assigner une string vide au innerHTML de userListUI
//TODO : Sur la variable snap on va utiliser un forEach pour parcourir le tableau avec une variable temporaire childSnap
//TODO : Dans le forEach : dans une variable key on va stocker childSnap.key
//TODO : Dans le forEach : dans une variable value on va stocker childSnap.val()
//TODO : Dans le forEach : dans une variable $li on va créer un element <li></li>
//TODO : Dans le forEach : dans le innerHTML de $li on lui assigne value.name
//TODO : Dans le forEach : Sur la $li on lui rajoute un attribut 'user-key', on lui assignera la valeur stockée dans key
//TODO : Dans le forEach : dans la userListUI on va placer $li
```

### Liste des Utilisateurs :

AAA

ZZZ

EEE

#### Auteur :

Jean-François Pech

#### Date création :

03/03/2023

#### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

#### Date révision :

10/03/2023

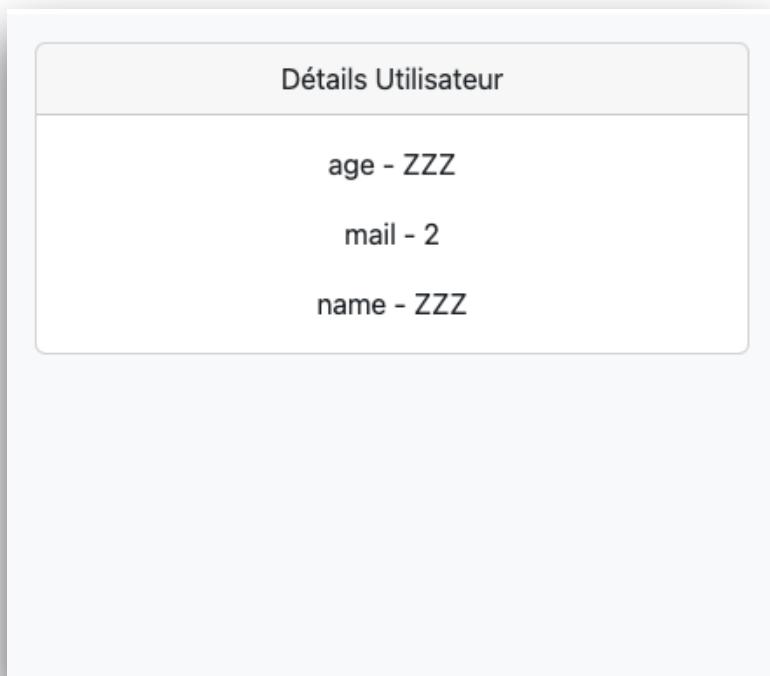


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : READ (Lecture d'un élément de la BDD : Afficher 1 utilisateur)

```
/* LIRE 1 USER
//TODO: Dans readUserData avant le append(), on va placer un addEventListener sur $li qui écoute « click » et lance la fonction userClicked
//? Ensuite dans la fonction userClicked on capte l'évènement (on s'en sert pour savoir qui on sélectionne)
//TODO: Dans la f° userClicked, Dans 1 variable userID, on va récupérer userID via event.target.getAttribute("user-key");
//TODO: Dans la f° userClicked, 1 variable userRef va faire référence à 1 utilisateur en particulier, on lui assigne dbRef,
//? on utilise la fonction child() pour viser le noeud "users/" concaténé avec userID
//TODO: Dans la f° userClicked, 1 variable userDetailUI récupère ma div avec user-detail
//TODO: Dans la f° userClicked, Ensuite sur userRef on utilise la fonction on("value", snap =>{ })
//TODO: Dans la f° userClicked, Dans la fonction =>, on va vider l'innerHTML de userDetailUI
//TODO: Dans la f° userClicked, Ensuite sur snap on va utiliser un forEach pour parcourir le cliché (snap) de notre BDD.
//TODO: Dans la f° userClicked dans le forEach, 1 variable $p créée un élément <p>
//TODO: Dans la f° userClicked dans le forEach, On remplit le innerHTML de $p avec childSnap.key et childSnap.val()
//TODO: Dans la f° userClicked dans le forEach, On rajoute $p dans notre userDetailUI
```

Quand on click sur un utilisateur dans la liste cela va récupérer son ID, pour lire la base de donnée et afficher les informations dans la div user-detail



**Auteur :**

Jean-François Pech

**Date création :**

03/03/2023

**Relu, validé & visé par :**

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

**Date révision :**

10/03/2023



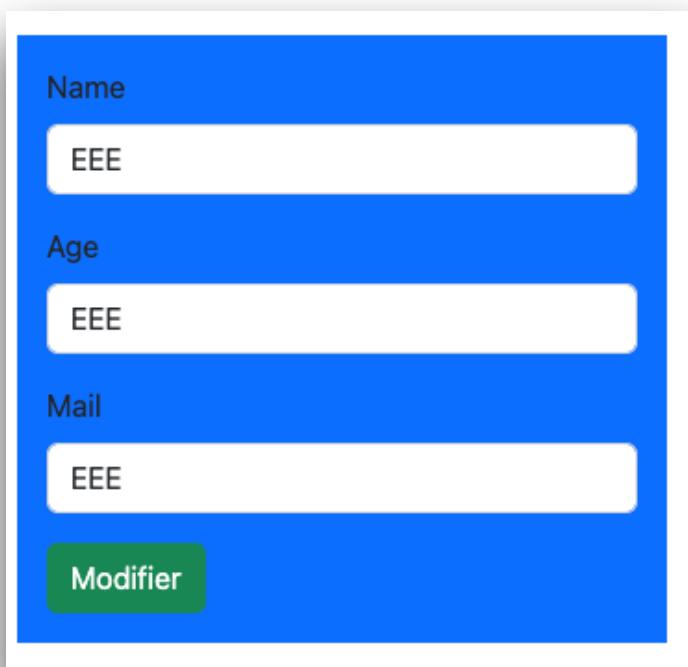
Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : READ (Lecture d'un élément de la BDD : Pré-remplir le formulaire d'un utilisateur)

Cette fonction est assez similaire à la précédente elle va lire les informations d'un utilisateur en base de données pour pré-remplir le formulaire de modification.

```
//*UPDATE (EDITER)
//! Dans la f° editButtonClicked
//TODO: on va modifier le display d formUserEditUI à block
//TODO: on va modifier le display du forUserUI à none
//TODO: Ensuite on va faire ceci :
//TODO: Une variable inputId qui récupère (querySelector) l'élément avec la classe .edit-userid
//TODO: A la valeur de inputId on assigne event.target.getAttribute("userid");
//TODO: Créer une variable userRef on lui assigne dbRef.child('users/' + inputId);
//TODO: Dans une variable editUserInputsUI, on récupère tous les éléments de classe edit-user-input (querySelectorAll ou autre)
//TODO: On va parcourir notre BDD avec userRef.on("value", snap => {
//TODO: dans la f° fléchée, Faire une boucle for qui parcourt les inputs editUserInputsUI,
//TODO: dans la f° fléchée dans la boucle, dans une variable key, on stock editUserInputsUI[i].getAttribute("data-key");
//TODO: dans la f° fléchée dans la boucle, Ensuite à chaque valeur de nos editUserInputsUI[i] on assigne snap.val()[key];
//TODO: En dehors de la fonction on(), Dans une variable saveBtn, on récupère notre bouton avec l id édit-user-btn
//TODO: En dehors de la fonction on(), Sur ce bouton on place un eventListener au click qui lance saveUserBtnClicked
```

Quand on clique sur le bouton update dans la liste des utilisateurs



### Auteur :

Jean-François Pech

### Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

### Date création :

03/03/2023

### Date révision :

10/03/2023

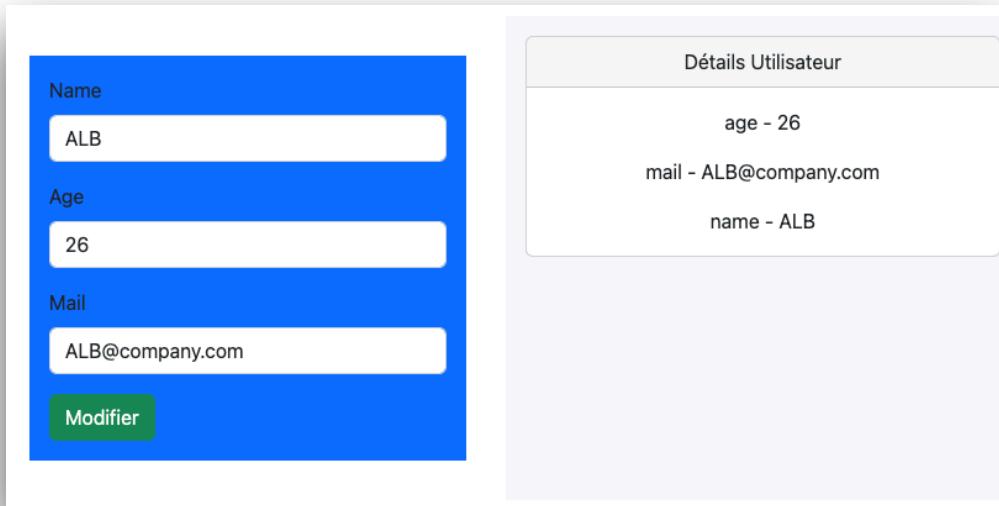


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Exercice : UPDATE : Sauvegarder les modifications d'un utilisateur

```
/*SAVE
//TODO : Dans la f° saveUserBtnClicked :
//TODO : Dans une variable userID on récupère la VALUE de l'input avec l'id "edit-userid"
//TODO : Dans une variable userRef on fait une référence à l'utilisateur dans la BDD
//TODO : Une variable editedUserObject qui est un objet vide
//TODO : Dans une variable editUserInputsUI on récupère TOUS les éléments html qui ont la classe "edit-user-input" (querySelectorAll)
//TODO : Ensuite on va faire une boucle forEach pour parcourir les editUserInputsUI
//TODO : Dans les param de forEach(), on lui passe une fonction qui a une variable textField en paramètre
//TODO : Dans cette fonction, dans le forEach on aura une variable key qui va stocker les attributs data-key de textField (getAttribute())
//TODO : Pour chaque clé (âge, name, email) on l'associe à notre nouvel utilisateur : editedUserObject[key] = textField.value
//TODO : Ensuite en dehors de la boucle, sur notre variable userRef on utilise la f° update en lui passant editedUserObject en paramètre
//TODO : Enfin on peut remettre le display à "none" de formUserEditUI et à "block" pour formUserUI|
```

Quand on click sur le bouton « modifier » de notre formulaire de modification d'un utilisateur, cela met à jours les données d'un utilisateur en temps réel



The screenshot illustrates a user modification process. On the left, a blue-themed form titled 'Name' contains three input fields: 'ALB' for Name, '26' for Age, and 'ALB@company.com' for Mail. Below these is a green 'Modifier' button. On the right, a modal window titled 'Détails Utilisateur' displays the updated information: 'age - 26', 'mail - ALB@company.com', and 'name - ALB'. This visualizes how changes made in the UI are reflected in the database.

## Exercice : DELETE (Supprimer un utilisateur)

```
/* SUPPRIMER
//TODO 1: Dans la f° readUserData, dans le forEach, juste avant $li.innerHTML = ...
//TODO 2: Dans la f° readUserData, dans le forEach, On va déclarer une variable deleteIconUI dans laquelle on va créer un élément span
//TODO 3: Dans la f° readUserData, dans le forEach, On va ensuite modifier la class de deleteIconUI en « delete-user »
//TODO 4: Dans la f° readUserData, dans le forEach, On va remplir le innerHTML de deleteIconUI avec un « X »
//TODO 5: Dans la f° readUserData, dans le forEach, deleteIconUI on lui rajoute un attribut « userid » qui prendra la valeur de key( via setAttribute)
//TODO 6: Dans la f° readUserData, dans le forEach, Enfin sur deleteIconUI on place un addEventListener qui écoute le click et lance la fonction deleteButtonClicked
//TODO 7: Créer une fonction deleteButtonClicked qui prend event en paramètre
//TODO 8: Dans la f° deleteButtonClicked, récupérer le userID via event.target.getAttribute
//TODO 9: Dans la f° deleteButtonClicked,
//TOD09-2: Faire une référence userRef à notre BDD directement sur le noeud de l'utilisateur qu'on a cliqué (référence à la table users + userID)
//TODO 10: Dans la f° deleteButtonClicked, utiliser la fonction remove sur userRef
```

On vise un utilisateur en particulier (comme dans les fonctions userClicked, editButtonClicked, et saveUserBtnClicked) pour le supprimer en temps réel de la base de données

<https://github.com/jefff404/cours-js/tree/25-firebase-cdn-js>

## Conclusion



Pour aller plus loin :

<https://bradfrost.com/blog/post/front-of-the-front-end-and-back-of-the-front-end-web-development/>

Des frameworks FrontEnd :

- [Angular](#)
- [Vue](#)
- [React](#)
- [Ember](#)

Framework SSR (Server Side Rendering) :

Qu'est ce que le SSR ? Le rendu côté serveur d'une page web ou Server Side Rendering (SSR) est une technique de développement web qui consiste à créer les pages html côté serveur pour les envoyer toutes faites au navigateur.

- [Nuxt](#)
- [Next](#)
- [Svelte](#)
- [Qwik](#)

Framework Mobile - Cross Platform :

- [Ionic](#)
- [React Native](#)

De la 3D - AR-VR ? :

- [Three.js](#)

Automatisation - Testing :

- [Mocha](#)
- [Jest](#)
- [Jasmine](#)
- [Cypress](#)
- [Puppeteer](#)

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE  
 Sophie POULAKOS  
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.