

JAVASCRIPT

Durée Totale du Module : 21H

Auteur :

Mathieu Mithridate

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

Durée Totale du Module : 21H

Présentation de Javascript

7

Langage de programmation	8
Multi Environnement	8
Orienté Objet	10
Web Dynamique	10
Algo / Syntaxe JS Moderne (ES6 et +)	13
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 \oplus / Incrémente / Assignement Composé	20
Exercice Calculs - Nombres	20
Solution possible	21
Chaînes de caractères	22
Exercice Phrase	23
Solution possible	25
Les tableaux	26

Auteur :

Mathieu Mithridate

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

Solution possible

Exercice array 2 les fonctions pour les tableaux

Solution possible

Exercice arrays 3

Solution possible

Fonctions / Fonctions Fléchée / return / param / param par defaut / scope

Function

Paramètre

Exercice Fonction 1

Solution

Return

Paramètre par défaut

Scope

Exercice : Quiz 1 Trouver le bug

Exercice : Quiz 2 Trouver le bug

Exercice : Moyenne

Solution

Opérateurs de comparaison / condition ternaire

Opérateurs de comparaison

Condition / opérateur ternaire

Condition IF ELSE

Exercice : If Else

Solution

Objets

hasOwnProperty

Exercice : Objects

Solution

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

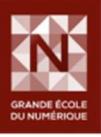
While

FOR

ForEach

For ... of

For ... in

L'ECOLE
REGIONALE DU
NUMERIQUE

26

27

28

29

30

31

32

32

32

33

34

35

35

36

36

37

37

37

38

39

39

40

41

41

42

43

44

44

44

45

46

46

47

48

49

50

Auteur :

Mathieu Mithridate

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

Exercice : boucles

Solution

Spread Operator

Point sur var let ou const

Exercice : Quizz Var

Exercice : Quizz let

Exercice : Quizz function-var

Solution function var

Exercice : Quizz if-var

Exercice Quizz : if-let

Solution if-let

Const

Gestion des erreurs

Try ... Catch

Gestion des exceptions avec Throw

Finally

Les Classes

Syntaxe

L'héritage

Exo Class IMC

Exo Class PME

Exo Class COMPTE BANCAIRES.

La guerre des langages

Asynchrone

API

Fetch()

Code Asynchrone

Async await

Exercice : Contacter une API

Then catch

Gestion des erreurs Response ET Promise

Javascript Modulaire

Web Workers



L'ECOLE
REGIONALE DU
NUMERIQUE

ADRAR
DIGITAL
ACADEMY



51

52

53

56

58

58

59

59

60

61

63

64

64

66

66

69

73

73

74

75

76

78

79

82

84

85

85

86

86

86

87

89

90

91

92

93

Auteur :

Mathieu Mithridate

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.

Présentation	94
Sélectionner des éléments du DOM	95
<code>getElement(s)By</code>	95
<code>QuerySelector()</code>	97
Placer des éléments du DOM	98
<code>insertBefore()</code>	98
<code>append() / appendChild()</code>	99
<code>removeChild()</code>	101
Créer - Modifier - Supprimer des éléments du DOM	102
<code>createTextNode()</code>	102
<code>createElement()</code>	103
Exercice : Créer des éléments - Afficher un profil utilisateur	104
Solution Possible	105
Solution possible	107
Modifier les attributs des éléments du DOM	108
Modifier attribut style	108
Modifier attribut class	109
Fonction classList	110
Modifier ou créer n'importe quel attribut	110
Gérer les événements du DOM	111
Exercice : réagir au click	112
Solution possible	113
Exercice réagir au click 2	114
Solution possible	115
Exo : réagir au click 3 (capter l'event)	116
Solution Possible	117
Exercice : réagir au focus et au Blur	118
Solution possible	120
Exercice : réagir au Load	121
Exercice : réagir à mouseleave	123
Solution Possible	124
Exercice : réagir au scroll	125
Solution possible	126

Auteur :

Mathieu Mithridate

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.



ADRAR
FORMATION

Exercice : réagir au clavier

Solution Possible

Exercice : Réagir au clavier 2

Web Storage

La fonction setItem()

La fonction getItem()

Exercice : LocalStorage

Solution Possible

Exercice DOM Settimeout

B.O.M

JS & Firebase

Set-up de firebase

Setup-JS

Setup-HTML

Setup-CSS

Exercice : CREATE : Ajouter un nouvel utilisateur

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

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

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

Exercice : UPDATE : Sauvegarder les modifications d'un utilisateur

Exercice : DELETE (Supprimer un utilisateur)

Conclusion



ADRAR
DIGITAL
ACADEMY

L'ECOLE
REGIONALE DU
NUMERIQUE



La Région
Occitanie
Pyrénées - Méditerranée



GRANDE ÉCOLE
DU NUMÉRIQUE



127

129

130

131

132

132

133

135

136

138

139

141

152

153

155

157

159

160

162

164

165

167

Auteur :

Mathieu Mithridate

Relu, validé & visé par :

Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date création :

03/03/2023

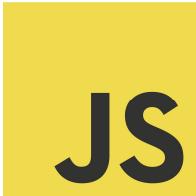
Date révision :

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.

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 :

Mathieu Mithridate

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.



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 coté 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 :

Mathieu Mithridate

Date création :

03/03/2023

Relu, validé & visé par :

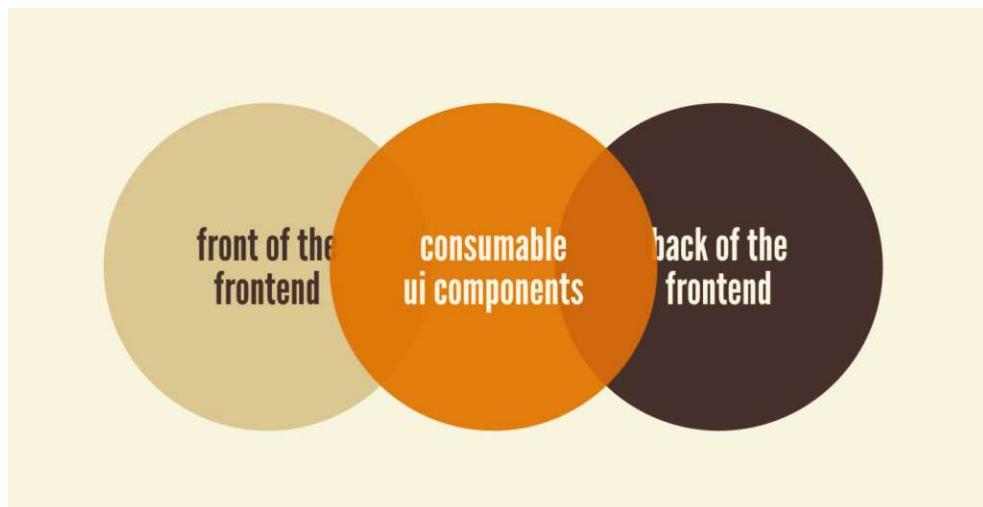
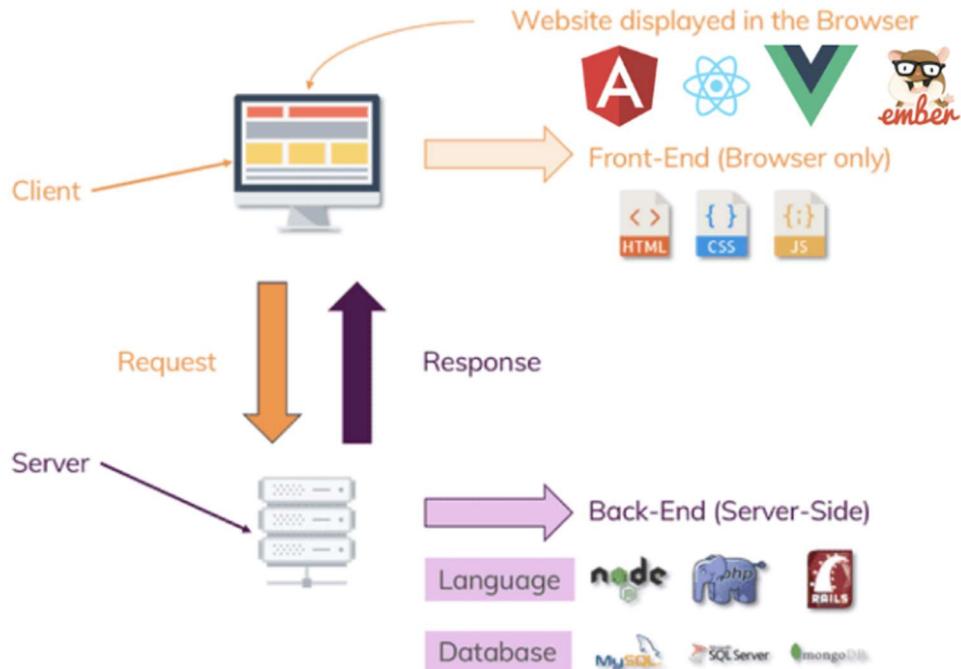
Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

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.



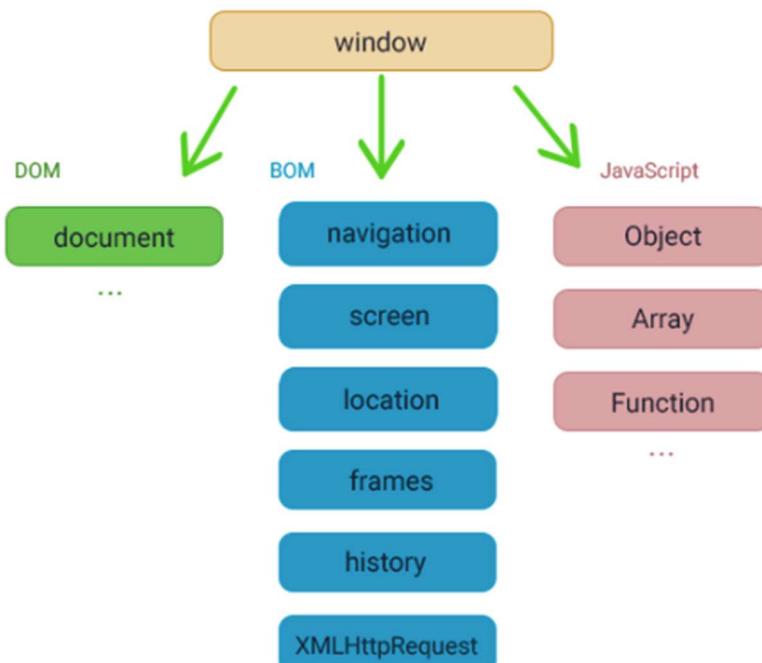
Orienté Objet

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.

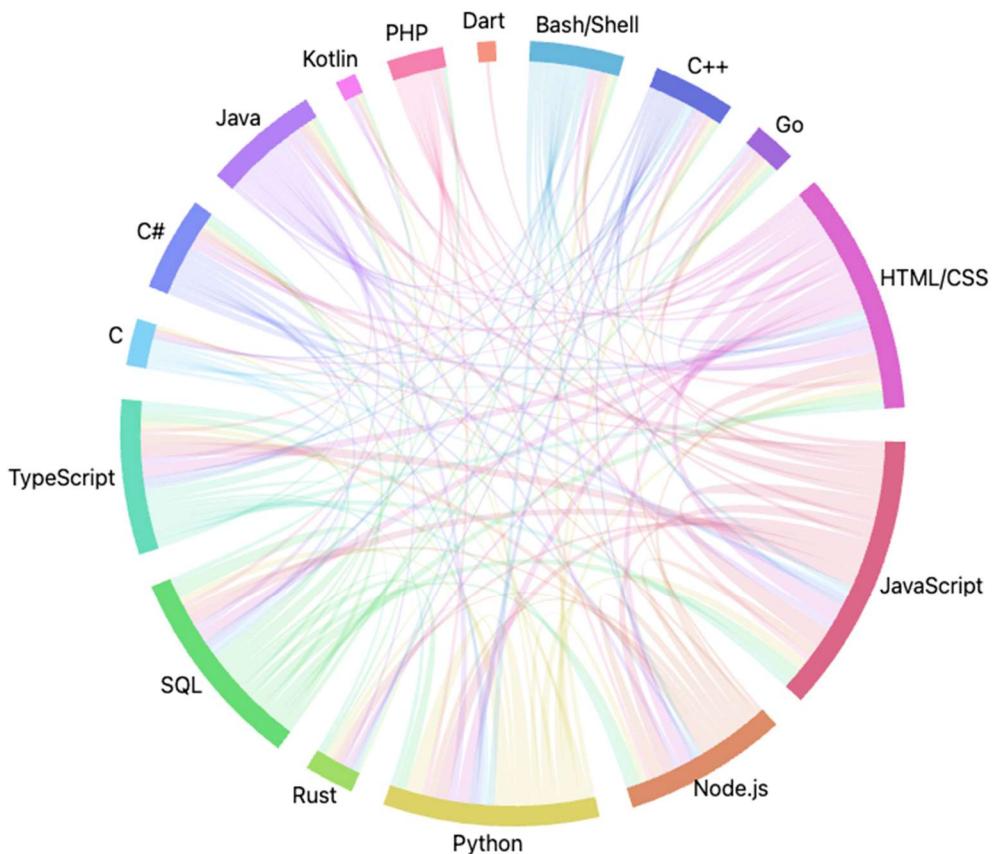


JS : Actuellement

Auteur :	Date création :		
Mathieu Mithridate	03/03/2023		
Relu, validé & visé par :	Date révision :		
<input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> Mathieu PARIS	10/03/2023	 TECH' CARE 	
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.			

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 :

Mathieu Mithridate

Relu, validé & visé par :

- Jérôme CHRETIENNE
- Sophie POULAKOS
- Mathieu PARIS

Date création :

03/03/2023

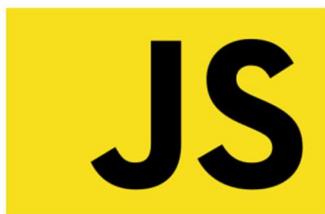
Date révision :

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.

2008



2021



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

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

Auteur : Mathieu Mithridate	Date création : 03/03/2023	Relu, validé & visé par : <input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> Mathieu PARIS	Date révision : 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.
---------------------------------------	--------------------------------------	---	--------------------------------------	---	---



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)

Auteur :

Mathieu Mithridate

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.

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 :

Mathieu Mithridate

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.

```
<!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 -->
  <script>console.log('Hello World')</script>
</body>
</html>
```

Côté Template

Async ou Defer

On peut aussi placer la balise script pour relier le fichier JS dans le haut du HTML, dans la balise <head>

```
<!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>
```

MAIS il faut ajouter l'attribut `async` ou `defer`

<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.

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 :

Mathieu Mithridate

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.

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

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 :	Date création :	
Mathieu Mithridate	03/03/2023	
Relu, validé & visé par :	Date révision :	
<input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> Mathieu PARIS	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.

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)
- Symbole (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 :

Mathieu Mithridate

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 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 :

Mathieu Mithridate

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.

Nombres / Calculs /Modulo ☺/ Incrémente / Assignement 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.

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

Auteur :

Mathieu Mithridate

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

```
/*
* ****
* 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
```

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



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 :

Mathieu Mithridate

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 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.

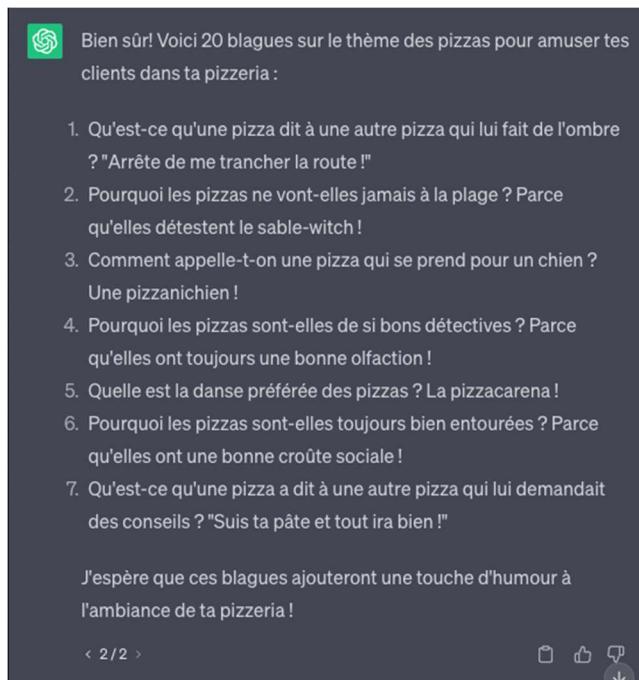
Définissez selon vous toutes les variables pertinentes qui résume la commande d'un utilisateur chez "La Pizzeria Raffinata"

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. »

Auteur :	Date création :			
Mathieu Mithridate	03/03/2023			
Relu, validé & visé par :	Date révision :			
<input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> Mathieu PARIS	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.				



L'ECOLE
REGIONALE DU
NUMERIQUE



ADRAR
DIGIT@L ACADEMY



Auteur :

Mathieu Mithridate

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 possible

```
/**  
* *****  
* 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'  
`;
```

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

Auteur :

Mathieu Mithridate

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.

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).

```
//! 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 variable 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
```

Exercice : Arrays

Auteur :	Date création :	
Mathieu Mithridate	03/03/2023	
Relu, validé & visé par :	Date révision :	
<input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> Mathieu PARIS	10/03/2023	 ADRAR DIGITAL ACADEMY

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 = 'JoSC';
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] = 'Y OLO';
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 :

Mathieu Mithridate

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 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
```

Exercice array 2 les fonctions pour les tableaux

Auteur :

Mathieu Mithridate

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 virgule pour ajouter plusieurs

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

données en utilisant une seule fois la fonction.

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

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

tableau :

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

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

Auteur :

Mathieu Mithridate

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 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
```

Exercice arrays 3

Auteur :

Mathieu Mithridate

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.

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

Solution possible

Auteur :

Mathieu Mithridate

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.



L'ECOLE
REGIONALE DU
NUMERIQUE



ADRAR
DIGITAL ACADEMY



Fonctions / Fonctions Fléchée / return / param / param par defaut / 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

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

Syntaxe de base pour déclarer une fonction

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

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

Paramètre

Les fonctions ont aussi un concept de paramètre, dans le cas où les instructions au sein de la fonction ont

```
/// 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);
```

besoin d'une variable extérieure. Ci-dessous une fonction qui va afficher en console une variable unNom
La variable num est définie directement lors de l'utilisation de la fonction.

Auteur : Mathieu Mithridate	Date création : 03/03/2023	Relu, validé & visé par : <input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> 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 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 :

Mathieu Mithridate

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.

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

Solution

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

- On peut aussi prévoir des fonctions nécessitant plusieurs paramètres comme ceci

Auteur :

Mathieu Mithridate

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 Uniquelement 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.

```
//! 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 ca
//! 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
//! resultat
let resultat = calculReturn(22,99);
console.log(resultat);
// ou executer la fonction quand on a besoin
console.log('Le résultat : ', calculReturn(22,99));
```

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

Paramètre par défaut

Une bonne pratique lorsque l'on écrit des fonctions (surtout en travail collaboratif), consiste à renseigner un

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

paramètre par défaut dans le cas où on oublie de renseigner un paramètre quand on exécute une fonction.

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.

```
// ? 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);
```

Une fonction va pouvoir utiliser des variables déclarées globalement mais le programme global ne peut pas utiliser une variable déclarée dans une fonction.

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

Exercice : Quiz 1 Trouver le bug

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

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  
    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 :

Mathieu Mithridate

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.

```
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);
```

Solution

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

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

```
/**  
* *****  
* 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");
```

un booléen (true ou false).

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.

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

```
//!-----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 ☐');
```

puis « : » et ce qui est return quand la condition n'est pas remplie

On peut aussi combiner plusieurs conditions avec les opérateurs

```
// ? 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');
```

|| (une condition OU une autre condition), && (une condition ET une autre condition)

Auteur :

Mathieu Mithridate

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.



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));
```

```
//! 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)
```

Exercice : If Else

Auteur :	Date création :		
Mathieu Mithridate	03/03/2023		
Relu, validé & visé par :	Date révision :		
<input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> Mathieu PARIS	10/03/2023	 TECH' CARE 	
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 :

Mathieu Mithridate

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.

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 objet n'ont pas de système d'indexation), c'est à nous de définir les propriétés (les clé) 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 créer la propriété)

Auteur :	Date création :	
Mathieu Mithridate	03/03/2023	
Relu, validé & visé par :	Date révision :	
<input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> Mathieu PARIS	10/03/2023	 ADRAR <small>FORMATION</small>

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'ECOLE
REGIONALE DU
NUMERIQUE

ADRAR
DIGITAL ACADEMY



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

```
// ? 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;
```

```
delete user.badges;
```

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 Fidjii'  
};  
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 : Mathieu Mithridate	Date création : 03/03/2023	Relu, validé & visé par : <input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> 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.
---------------------------------------	--------------------------------------	---	--------------------------------------	--

```
// 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 = 'Il à Cinéma!'
```

Solution

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

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

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 :

Mathieu Mithridate

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.

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	<u>app.js:14</u>
boucle FOR : Mission Alcatraz	<u>app.js:14</u>
boucle FOR : Attack Force	<u>app.js:14</u>

Auteur :

Mathieu Mithridate

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.

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

```
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));
```

fléchée).

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 :

Mathieu Mithridate

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.

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 :

Mathieu Mithridate

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.



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
};
```

```
// 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]}`);
}
```

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

Mathieu Mithridate

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° 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 :

Mathieu Mithridate

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 : 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 methode 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 parametre une
variable temporaire
(nom de la variable au choix)
//TODO cette fonction fléchée elle va modifier le innerHTML ou innerText de la variable
temporaire
```

Auteur :

Mathieu Mithridate

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 :

Mathieu Mithridate

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.

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?

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 :

Mathieu Mithridate

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.



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 :

Mathieu Mithridate

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.

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.

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

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

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

exemple :

Maintenant, nouvelObjet contient les mêmes propriétés que objetSource.

Auteur :

Mathieu Mithridate

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.

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

```
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
```

operator.

<https://github.com/jefff404/cours-js/tree/11-spread-operator>



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)

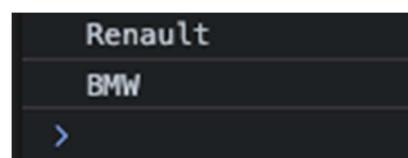
C'est plus de contraintes mais ça permet de garder une logique et un code plus propre
Avec sa versionES6 JS introduit des nouvelles manières de déclarer des variables

Exercice : Quizz Var

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

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

Que me répondez-vous ?
(Je vous jure cela fonctionne)



Auteur :

Mathieu Mithridate

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.



L'ECOLE
REGIONALE DU
NUMERIQUE

ADRAR
DIGITAL ACADEMY



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'
```

Exercice : Quizz function-var

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

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

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

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

Auteur :

Mathieu Mithridate

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 :

Mathieu Mithridate

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 : 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 :

Mathieu Mithridate

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.



L'ECOLE
REGIONALE DU
NUMERIQUE



ADRAR
DIGITAL ACADEMY



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

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

applications deviendra plus complexe

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

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

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

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

[app.js:48](#)

Auteur :

Mathieu Mithridate

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.

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

let voiture2 = 'Citroen';
```

On obtient dès lors des erreurs certes mais beaucoup plus précises.
Ici avec const : erreur voiture2 a déjà été déclarée

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

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

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

}
console.log(superCar);
```

Exercice Quizz : if-let

Auteur :

Mathieu Mithridate

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 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.

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

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

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

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

MyTracklist.track1 = 'félicien'
console.log(MyTracklist);
```

une de ses propriétés.

Au final en
et +) on va

```
>{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
```

JS moderne (ES6
privilégier

Auteur :	Date création :		
Mathieu Mithridate	03/03/2023		
Relu, validé & visé par :	Date révision :		
Jérôme CHRETIENNE Sophie POULAKOS Mathieu PARIS	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.			



ADRAR
FORMATION

l'utilisation de **let** et **const** pour créer des variables.



L'ECOLE
REGIONALE DU
NUMERIQUE



ADRAR
DIGITAL ACADEMY



Auteur :

Mathieu Mithridate

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.

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 la gestion d'erreurs au travers d'exemples.

Auteur :

Mathieu Mithridate

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.

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

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

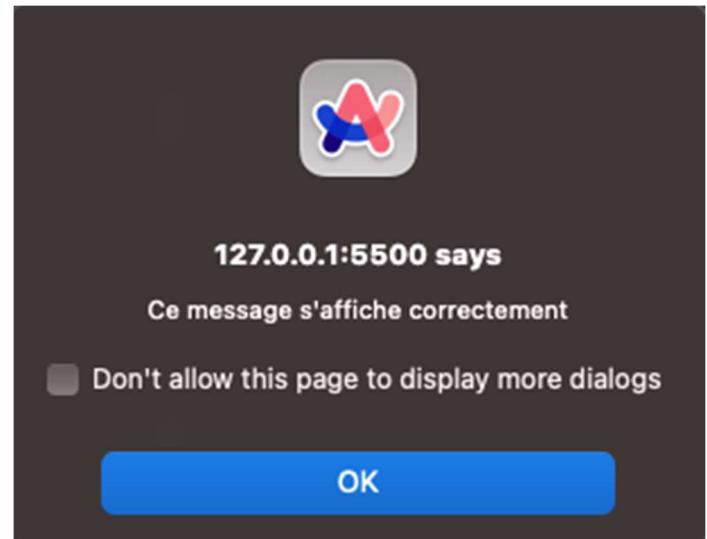
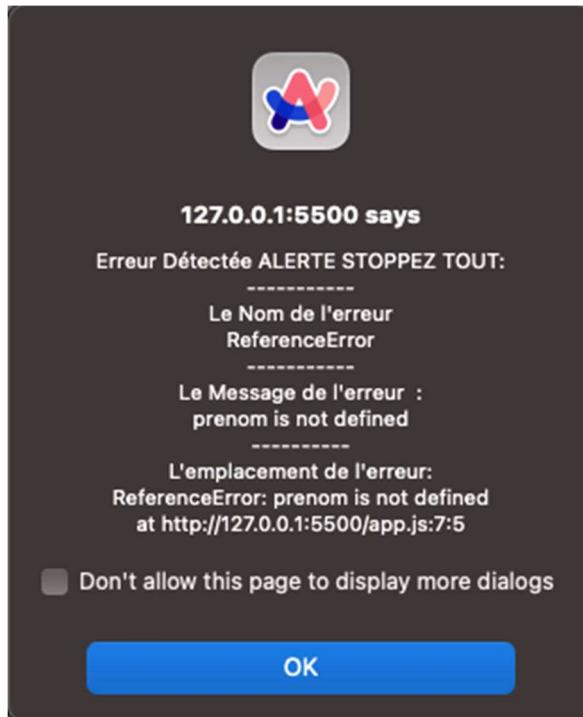
Erreur générée par la fonction alerte

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

Erreur générée par la variable prénom

```
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');
```

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

Mathieu Mithridate

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.

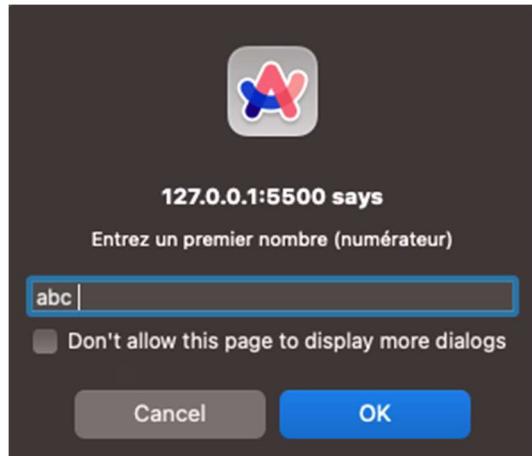
```
function division(){
    let x = prompt('Entrez un premier nombre (numérateur)');
    let y = prompt('Entrez un deuxième nombre (dénominateur)');

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

1 ère exception :



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

[app.js:29](#)

Auteur :

Mathieu Mithridate

Relu, validé & visé par :

Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date création :

03/03/2023

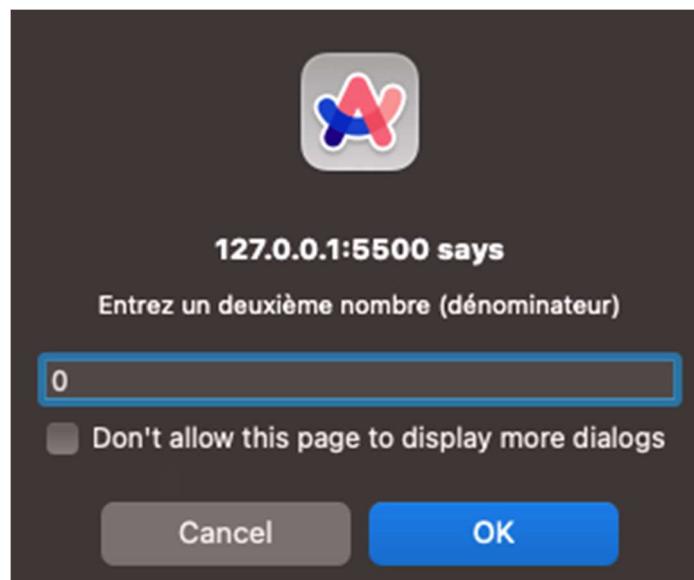
Date révision :

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.

2^e exception :



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

Auteur :

Mathieu Mithridate

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.

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.

Reprenons notre fonction division : ici on ne va plus exécuter la fonction division directement dans notre

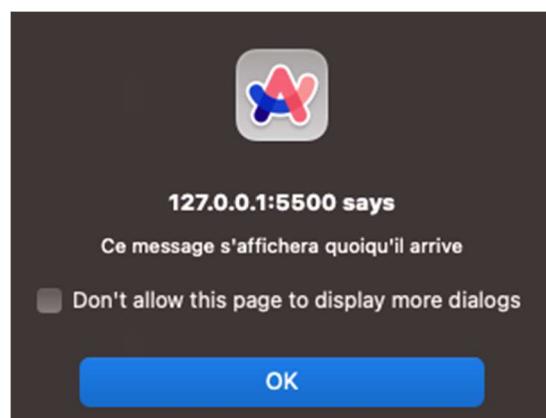
```
function division(){
    let x = prompt('Entrez un premier nombre (numérateur)');
    let y = prompt('Entrez un deuxième nombre (dénominateur)');

    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');
    }
}
```

programme on va essayer d'exécuter la fonction.



Auteur :	Date création :		
Mathieu Mithridate	03/03/2023		
Relu, validé & visé par :	Date révision :		
<input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> Mathieu PARIS	10/03/2023	  ADRAR <small>FORMATION</small>	
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 Classes

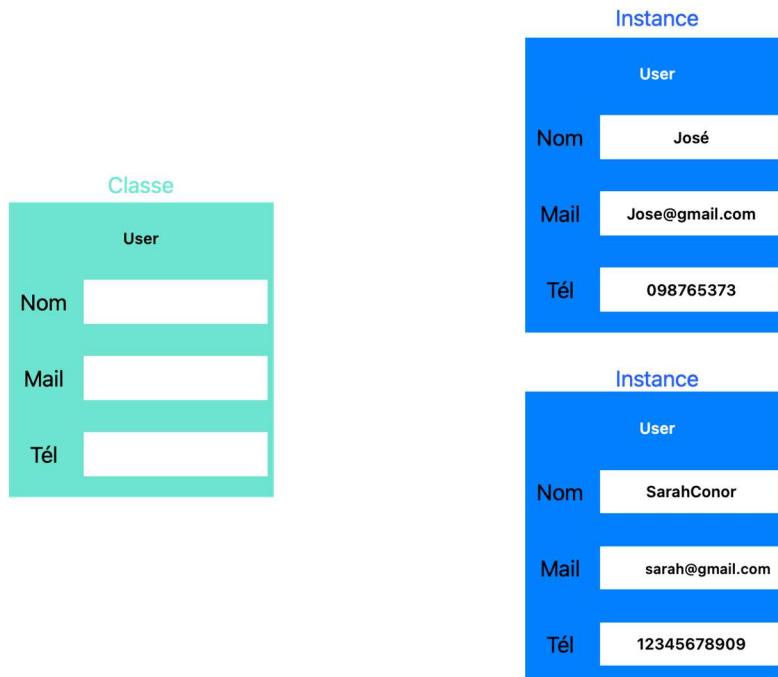
Pré requis : Les objets

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)



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.

```
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}`;
    }
}
```

Ici pour résumer on assigne aux propriété de notre classe les valeur qu'on va recevoir en paramètre
 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().

Auteur :

Mathieu Mithridate

Relu, validé & visé par :

Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date création :

03/03/2023

Date révision :

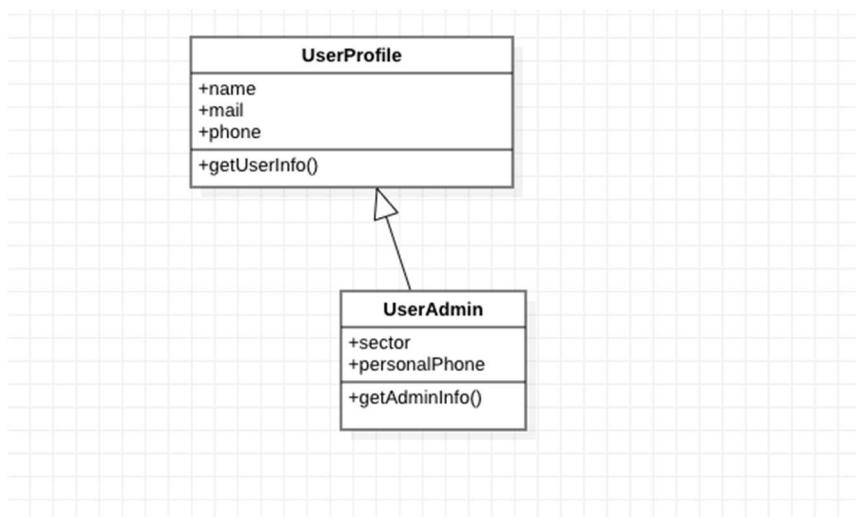
10/03/2023



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.

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()**.

```

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());

```

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

Auteur :

Mathieu Mithridate

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 navigateur

console du :
:

```
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
```

Auteur :

Mathieu Mithridate

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.

```
// /* 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("JOSiane de la Vega", 99, 1.55),
];
// *Boucle qui parcourt list pour utiliser display()
????????????((???????) ?? ????.????());
```

Programme Principal (en dehors de la classe)

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

Grace 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

Auteur :

Mathieu Mithridate

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.

- R = 300000 (trois cent mille)
- FF = 20000 (vingt mille)
- FA = 50000 (cinquante mille)
- N = 12
- XXX = 90%

En
du

console

```
-----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
```

navigateur :

Auteur :

Mathieu Mithridate

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 :

Mathieu Mithridate

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 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://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>

```
// 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 décrit 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 ?)
??????;
```

Auteur :

Mathieu Mithridate

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.

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) :

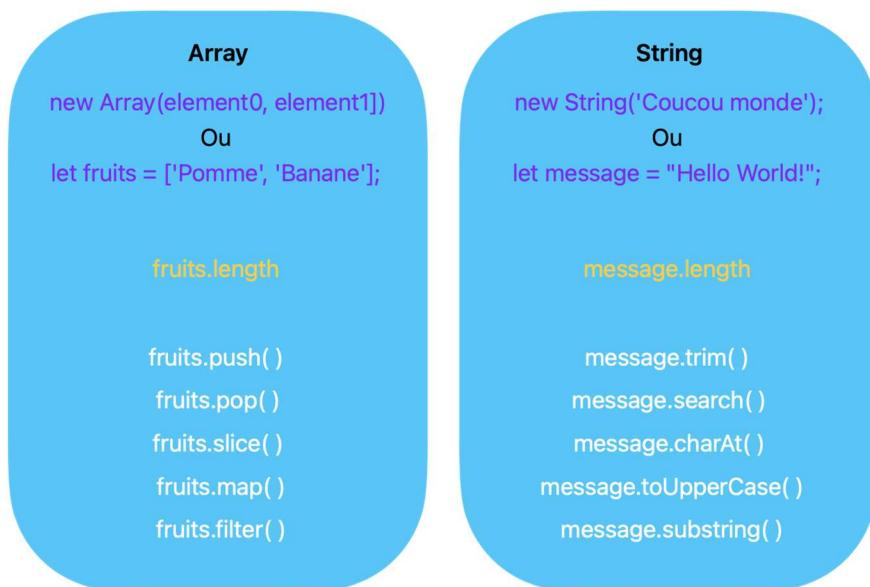
On peut déclarer soit en utilisant le constructor `new Array()`, mais il faut penser à stocker dans une variable, nativement dans JS pour chaque objet on aura des propriétés de base, ici `length` commun à plusieurs type et JS propose aussi des fonctions de base, utiles pour manipuler chaque type d'objets.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

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

Jaune : exemple d'une propriété

Blanc : des

fonctions de bases



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 :

Mathieu Mithridate

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.

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.

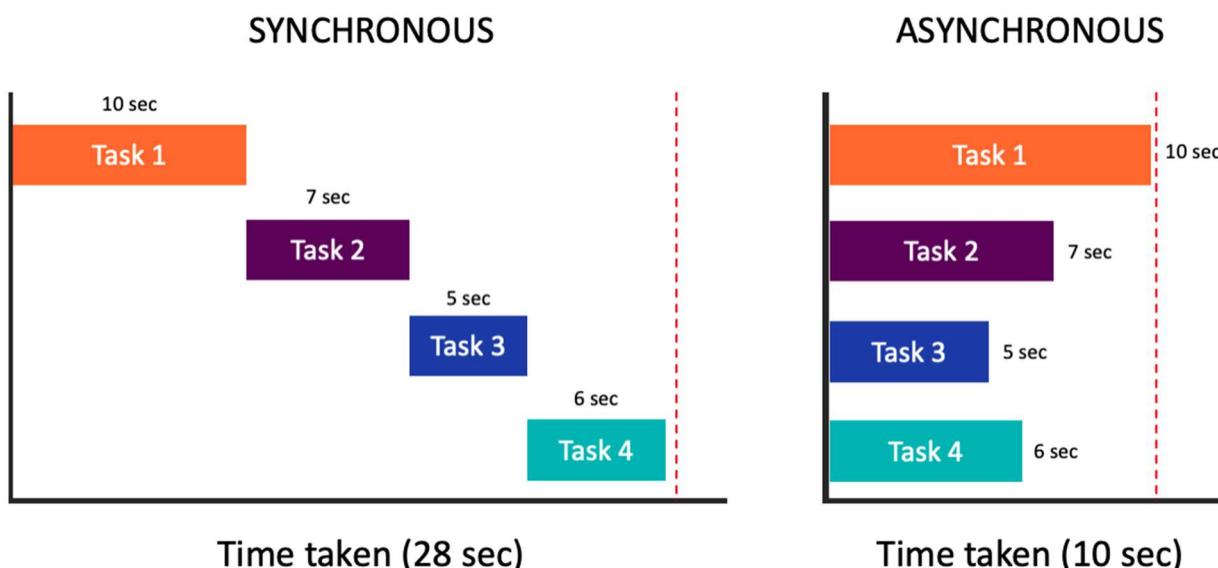
l'API Fetch et sa méthode `fetch()` qui correspondent à la "nouvelle façon" d'effectuer des requêtes HTTP.

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 :

Mathieu Mithridate

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.

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 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

```
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();
```

réponse en objet javascript (plus facile à manipuler), idem on précise que c'est en mode **await**.
Prenez le réflexe de toujours se référence à la documentation officielle des API

Auteur :

Mathieu Mithridate

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.



L'ECOLE
REGIONALE DU
NUMERIQUE



ADRAR
DIGITAL ACADEMY



```
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: (...)

  ▶ 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
```

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

Mathieu Mithridate

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.

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

```
// //** METHODE 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();
```

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

Fetch API

52.52

Auteur :

Mathieu Mithridate

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 réponse en accédant au propriétés de l'objet rappelez vous :
Ci dessous le code permet de gérer dès la réponse une erreur si l'url du fetch est invalide.

```
▼ 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
```

C'est mieux pour travailler en collaboratif et assurer un aspect **qualitatif** de votre code

```
/** 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 de la réponse
      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 :

Mathieu Mithridate

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.



ADRAR DIGITAL ACADEMY



Javascript Modulaire

Import export
Type module

Auteur :

Mathieu Mithridate

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.



ADRAR
FORMATION

Web Workers



ADRAR
DIGITAL
ACADEMY

L'ECOLE
REGIONALE DU
NUMERIQUE

**Auteur :**

Mathieu Mithridate

Relu, validé & visé par :

Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023

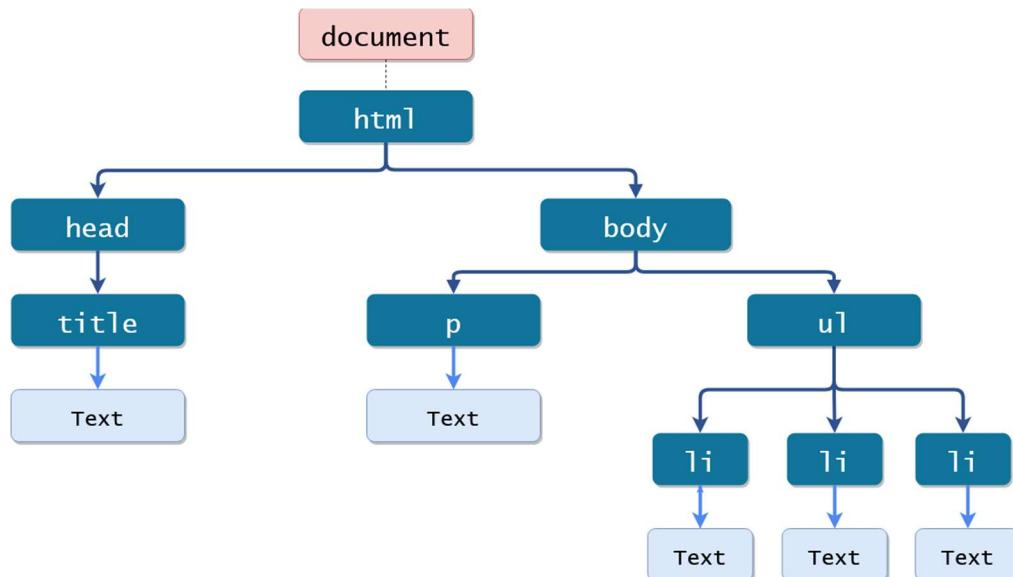


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.



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 :

Mathieu Mithridate

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.

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).

```
//? 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);
```

getElementsByName() - getElementsByClassName()

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

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

```
La HTMLCollection ▾HTMLCollection(10) ⓘ app.js:10
  ▷0: p
  ▷1: p.super
  ▷2: p
  ▷3: p
  ▷4: p
  ▷5: p#special
  ▷6: p
  ▷7: p
  ▷8: p
  ▷9: p
  ▷special: p#special
  length: 10
  ▷[[Prototype]]: HTMLCollection
```

paragraphes.

Auteur :	Date création :	
Mathieu Mithridate	03/03/2023	
Relu, validé & visé par :	Date révision :	
<input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> Mathieu PARIS	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.		

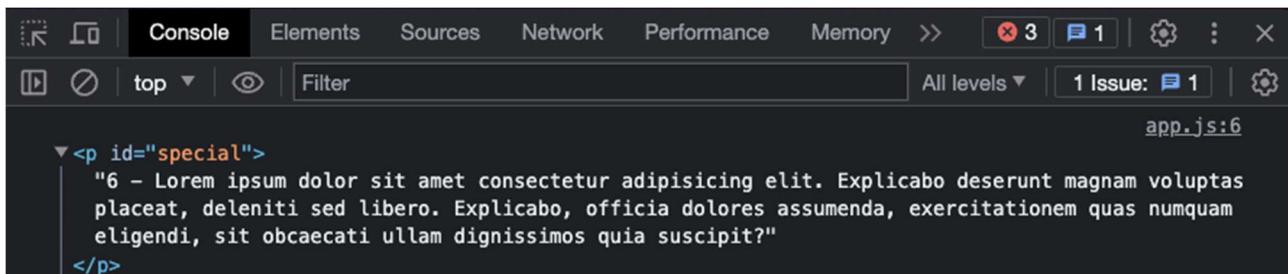


getElementById()

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);
```



Auteur : Mathieu Mithridate	Date création : 03/03/2023	Relu, validé & visé par : <input checked="" type="checkbox"/> Jérôme CHRETIENNE <input checked="" type="checkbox"/> Sophie POULAKOS <input checked="" type="checkbox"/> 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)

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

querySelector()

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

sélection par l'id (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);
```

Sélection par la classe (avec . comme en css)

querySelectorAll()

```
//? Une fonction type querySelector pour récupérer TOUS les éléments 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);
console.log('allParagraphes mais on prend le 2e',allParagraphes[1]);
```

Pour récupérer plusieurs éléments (par le nom de balise ou par leurs class css) dans une NodeList
<https://github.com/jefff404/cours-js/tree/20-dom>

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

Auteur :	Date création :		
Mathieu Mithridate	03/03/2023		
Relu, validé & visé par :	Date révision :		
Jérôme CHRETIENNE	10/03/2023		
Sophie POULAKOS			
Mathieu PARIS			



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.

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 , le

```
//! 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);
```

<body>, etc...), prend en compte 2 éléments Html en paramètre pour placer l'un avant l'autre

Page sans JS :

Page avec 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?

réation :
03/03/2023
évision :
10/03/2023



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?

iffusion ou document dé que ce expresse,

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...)

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

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

```
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 crée 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?

D.O.M (Placer des éléments)

Document Object Model

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



L'ECOLE
REGIONALE DU
NUMERIQUE



ADRAR
DIGITAL ACADEMY

**Auteur :**

Mathieu Mithridate

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()

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

Une fonction qui permet de retirer UN élément HTML du DOM
Ici on sur le body on supprime le 10è paragraphe

Auteur :

Mathieu Mithridate

Relu, validé & visé par :

Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



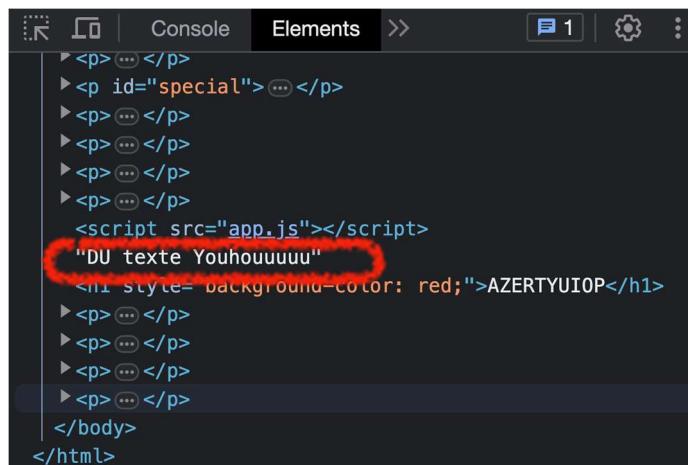
Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Créer - Modifier - Supprimer des éléments du DOM

`createTextNode()`

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

Une fonction pour créer du texte brut dans une page web



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

createElement()

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

```
///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
```

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

Ne pas oublier de
crées dans la page web



placer les éléments

Exercice : Créer des éléments - Afficher un profil utilisateur

```
//! 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'
};
```

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



John delavega
john.doe@example.com
25
08/02/1989
true

Auteur :

Mathieu Mithridate

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 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 ???????

//! 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

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

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 :

Mathieu Mithridate

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 possible

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

Auteur :

Mathieu Mithridate

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.

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';
```

Et si on

vérifie



via

Auteur :	Date création :	
Mathieu Mithridate	03/03/2023	
Relu, validé & visé par :	Date révision :	
Jérôme CHRETIENNE Sophie POULAKOS Mathieu PARIS	10/03/2023	
 TECH' CARE	 ADRAR	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.