

JAVASCRIPT

Durée Totale du Module : 21H

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

JAVASCRIPT

Durée Totale du Module : 21H

Présentation de Javascript	7
Langage de programmation	8
Multi Environnement	8
Orienté Objet	10
Web Dynamique	10
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
Setup Avec VITE	16
Les Variables	17
Anatomie d'une variable	17
Déclarer une variable et assigner une valeur	17
Plusieurs types de variable	18
Exercice :	18
Solution possible	19
Nombres / Calculs /Modulo @/ Incrémenter / Assignment Composé	20
Exercice Calculs - Nombres	20
Solution possible	21
Chaînes de caractères	22
Exercice Phrase	23

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

Solution possible	24
Les tableaux	25
Exercice : Arrays	25
Solution possible	26
Exercice array 2 les fonctions pour les tableaux	27
Solution possible	28
Exercice arrays 3	29
Solution possible	30
Fonctions / Fonctions Fléchée / return / param / param par défaut / scope	31
Function	31
Paramètre	31
Exercice Fonction 1	32
Solution	33
Return	34
Paramètre par défaut	34
Scope	35
Exercice : Quiz 1 Trouver le bug	35
Exercice : Quiz 2 Trouver le bug	36
Exercice : Moyenne	36
Solution	37
Opérateurs de comparaison / condition ternaire	38
Opérateurs de comparaison	38
Condition / opérateur ternaire	39
Condition IF ELSE	40
Exercice : If Else	40
Solution	41
Objets	42
hasOwnProperty	43
Exercice : Objects	43
Solution	44
Boucle While / For / ForEach / For ...of / For ...in / map	45
While	45

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

FOR	46
ForEach	47
For ... of	48
For ... in	49
Convertir des objets en tableaux	50
Exercice : boucles	51
Solution	52
Spread Operator	55
Point sur var let ou const	57
Exercice : Quizz Var	57
Exercice : Quizz let	58
Exercice : Quizz function-var	58
Solution function var	59
Exercice : Quizz if-var	60
Exercice Quizz : if-let	62
Solution if-let	63
Const	63
Gestion des erreurs	64
Try ... Catch	64
Gestion des exceptions avec Throw	67
Finally	70
Les Classes	71
Syntaxe	72
L'héritage	73
Exo Class IMC	75
Exo Class PME	76
Exo Class COMPTES BANCAIRES.	78
La guerre des langages	80
Asynchrone	81
API	81
Fetch()	82
Code Asynchrone	82

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

Async await	83
Exercice : Contacter une API	85
Then catch	86
Gestion des erreurs Response ET Promise	87
Javascript Modulaire	88
Web Workers	89
D.O.M	90
Présentation	90
Sélectionner des éléments du DOM	91
getElement(s)By	91
querySelector()	93
Placer des éléments du DOM	94
insertBefore()	94
append() / appendChild()	95
removeChild()	96
Créer - Modifier - Supprimer des éléments du DOM	97
createTextNode()	97
createElement()	98
Exercice : Créer des éléments - Afficher un profil utilisateur	99
Solution Possible	100
Solution possible	102
Modifier les attributs des éléments du DOM	103
Modifier attribut style	103
Modifier attribut class	104
Fonction classList	105
Modifier ou créer n'importe quel attribut	105
Gérer les évènements du DOM	106
Exercice : réagir au click	107
Solution possible	108
Exercice réagir au click 2	109
Solution possible	110
Exo : réagir au click 3 (capter l'événement)	111

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

Solution Possible	112
Exercice : réagir au focus et au Blur	113
Solution possible	114
Exercice : réagir au Load	115
Exercice : réagir à mouseleave	117
Solution Possible	118
Exercice : réagir au scroll	119
Solution possible	120
Exercice : réagir au clavier	121
Solution Possible	123
Exercice : Réagir au clavier 2	124
Web Storage	125
La fonction setItem()	126
La fonction getItem()	126
Exercice : LocalStorage	127
Solution Possible	129
Exercice DOM Settimeout	130
B.O.M	132
JS & Firebase	133
Set-up de firebase	134
Setup-JS	141
Setup-HTML	143
Setup-CSS	144
Exercice : CREATE : Ajouter un nouvel utilisateur	145
Exercice : READ (Lecture de la BDD : Tous Les Utilisateurs)	146
Exercice : READ (Lecture d'un élément de la BDD : Afficher 1 utilisateur)	147
Exercice : READ (Lecture d'un élément de la BDD : Pré-remplir le formulaire d'un utilisateur)	148
Exercice : UPDATE : Sauvegarder les modifications d'un utilisateur	149
Exercice : DELETE (Supprimer un utilisateur)	150
Conclusion	151

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

Présentation de Javascript



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 langage spécifique des OS.

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

Un langage de programmation

Multi Environnement

Orienté Objet

Permettant de rendre des pages web dynamiques

10 Jours plus tard naissait Javascript...

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

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

Langage de programmation

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

Multi Environnement

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

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



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

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

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

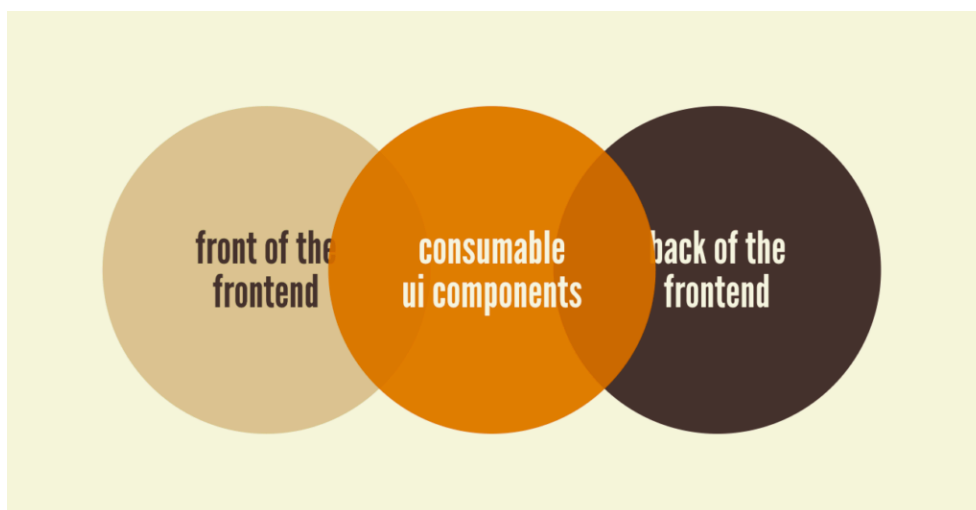
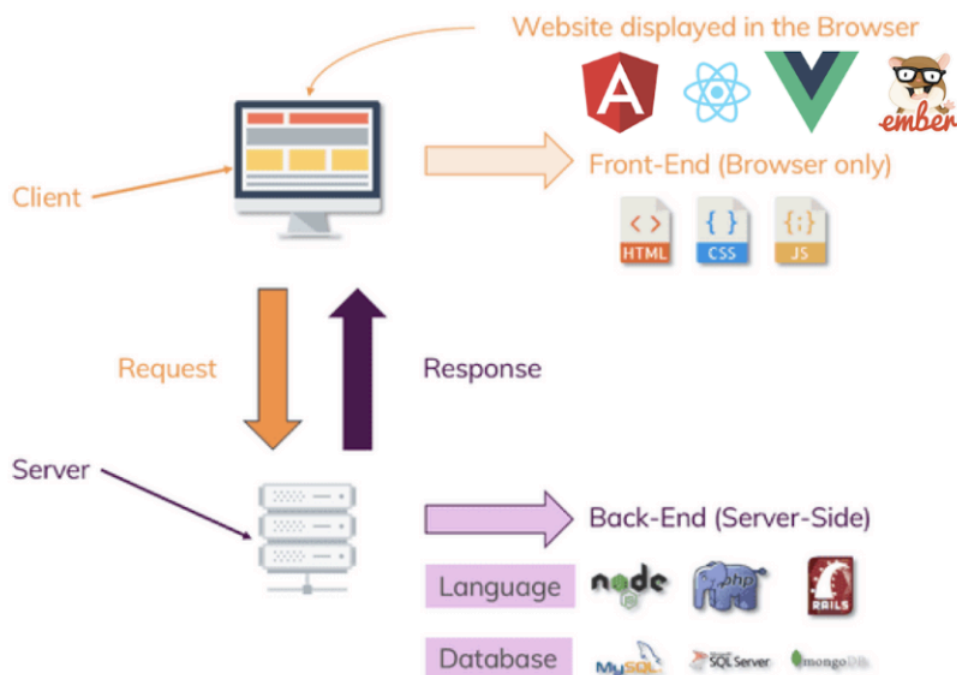
☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

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.



Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

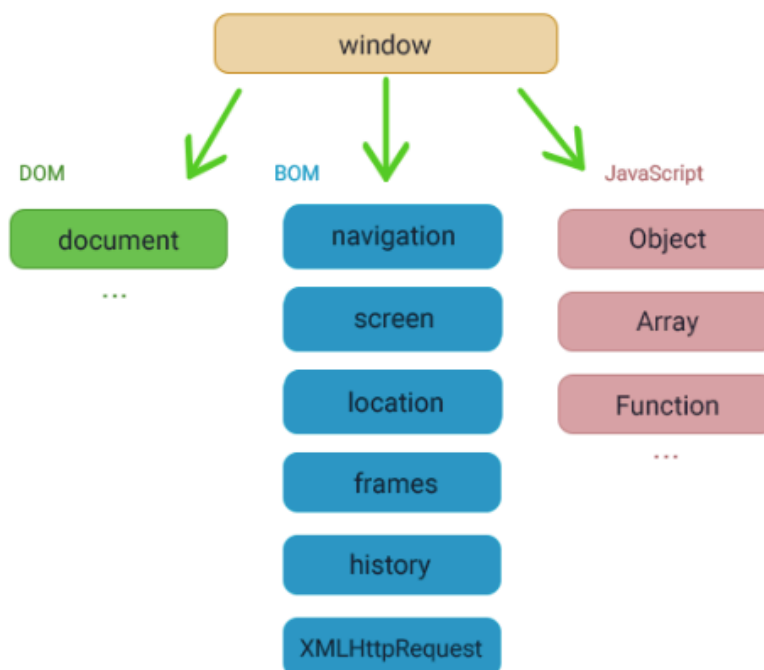
Orienté Objet

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

Web Dynamique

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

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



Auteur :

Jean-François Pech

Relu, validé & visé par :

☑ Jérôme CHRETIENNE
☑ Sophie POULAKOS
☑ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023

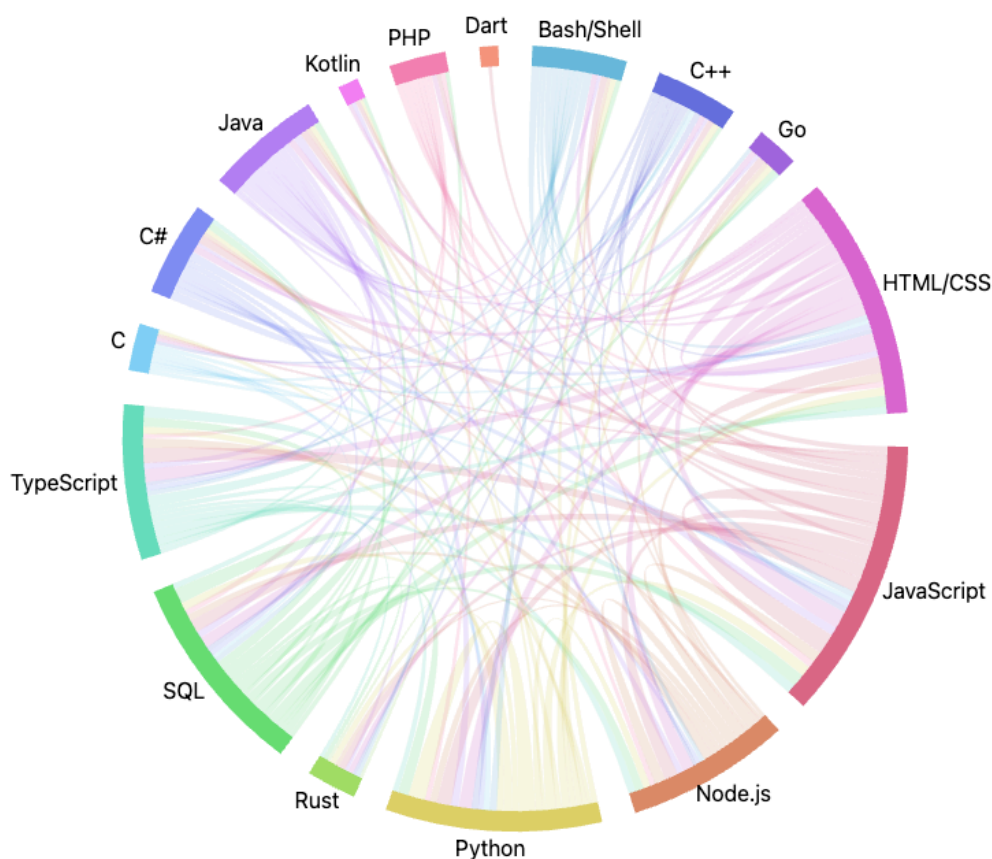


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

JS : Actuellement

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

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



Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



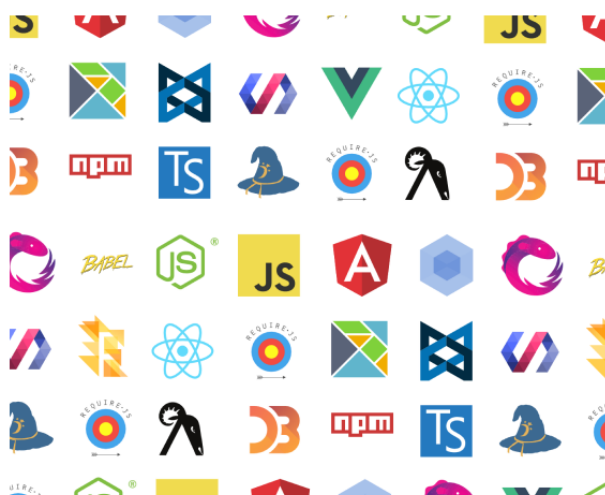
Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

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

2008



2021



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

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

Algo / Syntaxe JS Moderne (ES6 et +)

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

Amélioration de la DX



Visual Studio Code : super paramétrable, gratuit, plein d'extensions

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

Alternatives :

Webstorm

Fleet, etc..

En ligne

codepen

Stackblitz, etc...

Extensions vscode



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



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)

LiveServer



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 :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

Setup de Javascript

Il nous faut un éditeur de code, par exemple Visual Studio Code et pour plus de confort une extension permettant de simuler un serveur de développement Live Server

Et bien entendu nous aurons besoin d'un navigateur internet.(basé sur chromium, Firefox, Safari, etc...)

Setup Classique

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

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

Et un petit script dans le fichier Javascript

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

Auteur :

Jean-François Pech

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.

Côté Template

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

Async ou Defer

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

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

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

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

Auteur :

Jean-François Pech

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 : 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.

Setup Avec VITE

Check NodeJS pour avoir accès à NPM
Ou pnpm ou Yarn ou bun etc.. bref un package manager

Terminal (CD) se déplacer dans un endroit (là ou on met tous nos dossier d'app)

Faire npm

Auteur :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



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

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 :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



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

Plusieurs types de variable

Le dernier standard ECMAScript définit 8 types de données :
Sept types de données primitifs:

- Booléen (true / false)
- Null
- Undefined
- Number
- BigInt (proposition pour ES2020)
- String (chaîne 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 :

Jean-François Pech

Relu, validé & visé par :

- ☒ Jérôme CHRETIENNE
- ☒ Sophie POULAKOS
- ☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



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

Solution possible

<https://github.com/jeff404/cours-js/tree/2-variables>

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

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

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

Auteur :

Jean-François Pech

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 ☺/ Incrémenter / Assignment Composé

Bien entendu, une des utilités principales des langage 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 une variable compteur et utiliser tous les opérateur d'assignement composé.

Auteur :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



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

Solution possible

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

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

Auteur :

Jean-François Pech

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.

Chaînes de caractères

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

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

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

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

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

(Alt gr + 7)

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

Auteur :

Jean-François Pech

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.

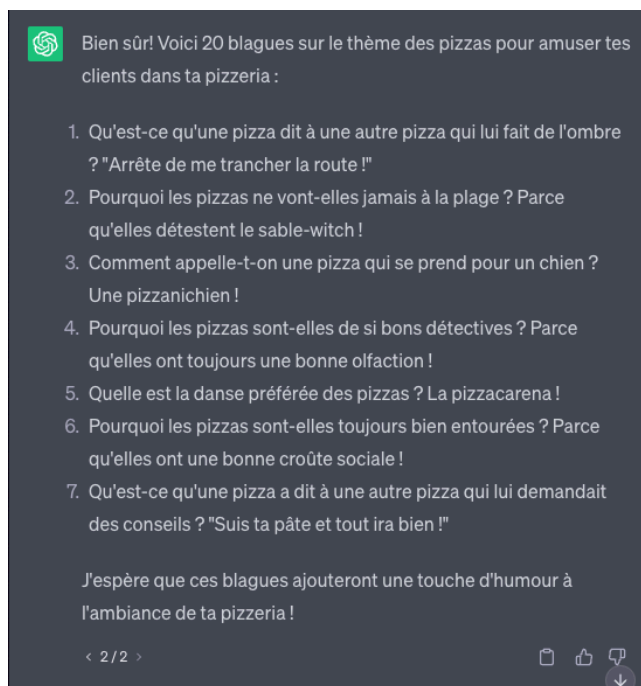
Définissez selon vous toutes les variables pertinentes qui résume la commande d'un utilisateur chez "La Pizzeria Raffinata"

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



« 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 :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



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

Solution possible

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

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

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

Les tableaux

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

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

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

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

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

```
mesNombres[2]
```

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

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

Exercice : Arrays

```
#!/ EXO 4 ARRAYS
//TODO: Créer 1 variable pour un nom,
//TODO: Créer une variable pour un âge,
//TODO: Créer une variable passions qui est un tableau qui contient 2 chaînes de
caractères (au choix)
//TODO: Puis créer une variable tabUser qui est un tableau qui contient les 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
```

Auteur :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



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

Solution possible

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

Auteur :

Jean-François Pech

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 array 2 les fonctions pour les tableaux

```
#!/EX0 4.3 Ajout f° .push()  
//TODO : créer un nouveau tableau qui contient des trucs  
//TODO : allez se renseigner la f° push()  
//TODO : utiliser push pour ajouter un nouveau truc au tableau  
//TODO: On affiche en console ce tableau
```

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

Solution possible

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

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

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

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

```
Après Ajout :                               app.js:37  
▶ (6) [10, 10, 10, 'Cortex', 92, 'Les Pyramides']  
  
suppression :                               app.js:42  
▶ (5) [10, 10, 10, 'Cortex', 92]
```

Auteur :

Jean-François Pech

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 arrays 3

```
//! EXO 4.4 ARRAYS R  cap  
// TODO: Cr  er 2 variables leNom et lePr  nom  
//TODO: Cr  er un tableau laPhrase et on y ajoute via push, Le nom ,Le pr  nom Les  
initiales  
//TODO: Afficher le tableau dans la console le nom le pr  nom et les initiales
```

Auteur :

Jean-Fran  ois Pech

Date cr  ation :

03/03/2023

Relu, valid   & vis   par :

☒ J  r  me CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date r  vision :

10/03/2023



Toute reproduction, repr  sentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque proc  d   que ce soit est interdite sans l'autorisation expresse,   crite et pr  alable de l'ADRAR.

Solution possible

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

Auteur :

Jean-François Pech

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.

Fonctions / Fonctions Fléchée / return / param / param par défaut / scope

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

Function

Syntaxe de base pour déclarer une fonction

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

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

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

Paramètre

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

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

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

Auteur :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



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

Exercice Fonction 1

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

Auteur :

Jean-François Pech

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

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

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

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

Auteur :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



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

Return

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

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

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

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

Paramètre par défaut

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

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

Auteur :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



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

Scope

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

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

Exercice : Quiz 1 Trouver le bug

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

✖ ▶ Uncaught ReferenceError: wtf is not defined app.js:71
 at app.js:71:13

Auteur :

Jean-François Pech

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 : Quiz 2 Trouver le bug

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

✖ ▶ Uncaught ReferenceError: lesNews is not defined [app.js:79](#)
at functionBugParent ([app.js:79:17](#))
at [app.js:86:1](#)

Exercice : Moyenne

```
//! EX0 5.2 : La moyenne de 2 notes  
//TODO: Créer une fonction qui calcule la moyenne de 2 notes  
//TODO: Afficher le résultat en console
```

Auteur :

Jean-François Pech

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.