

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.

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 :

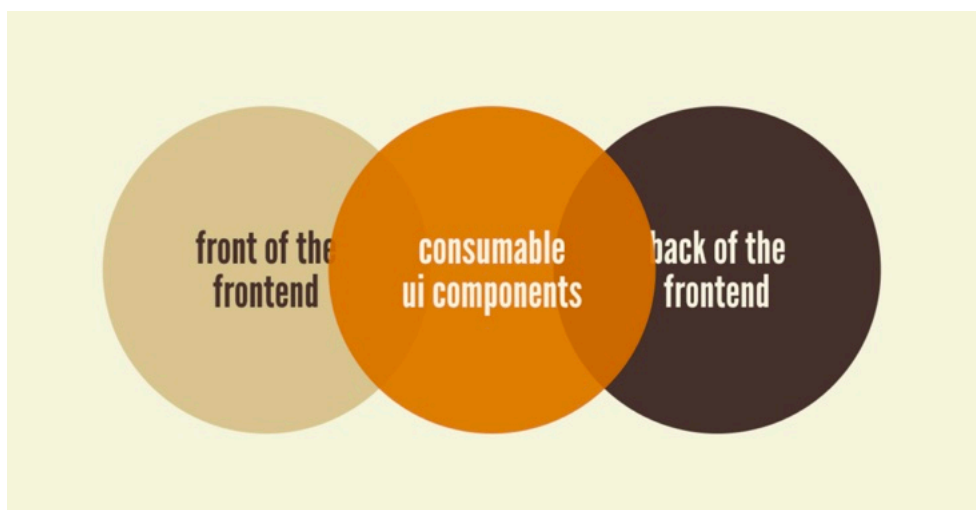
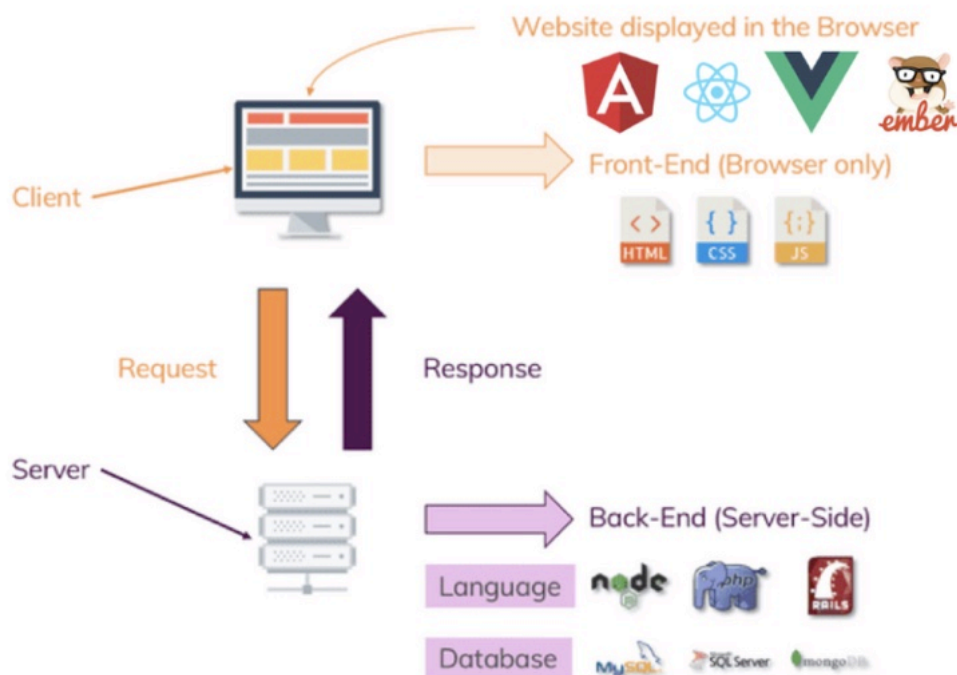
☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

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.

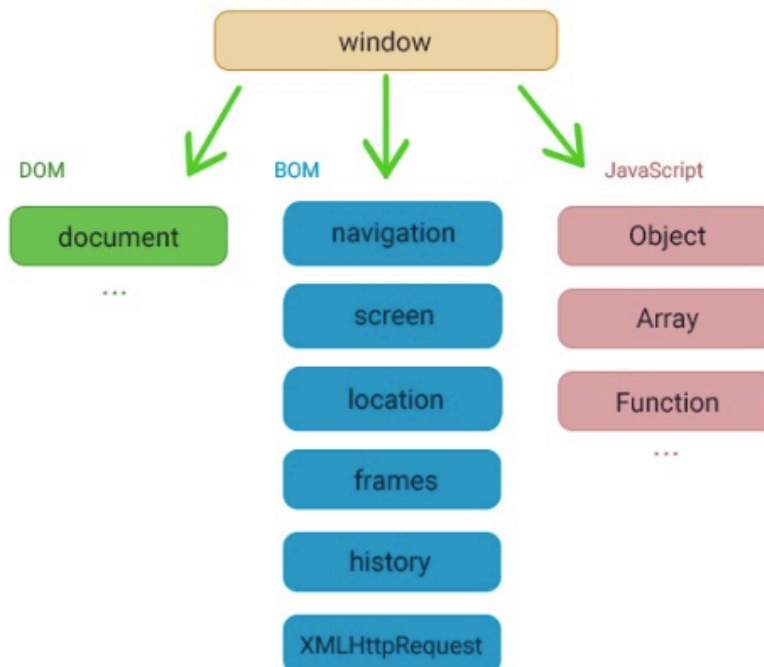
Orienté Objet

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

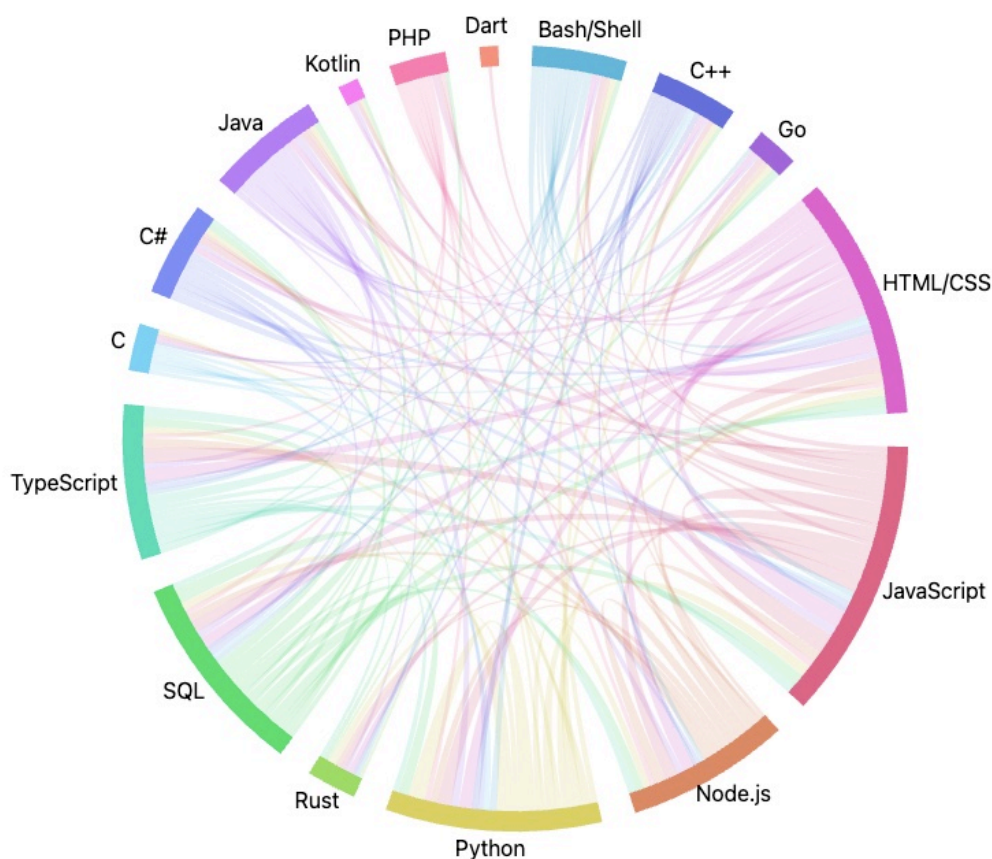


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

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



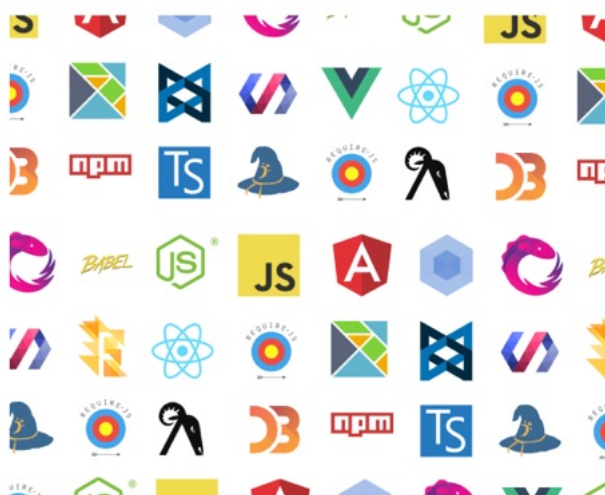
Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

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/>
<https://2023.stateofjs.com/en-US>

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 ou Five Server



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.

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 Avec VITE

<https://vitejs.dev/guide/#scaffolding-your-first-vite-project>

Pré requis :

NodeJS d'installé (pour pouvoir utiliser son gestionnaire de package avec la commande npm dans le terminal)

Ou pnpm ou Yarn ou bun etc.. bref un package manager

Terminal (CD) se déplacer dans un endroit (là ou on met tous le dossier de l'app)

Dans votre OS vous pouvez vous déplacer dans le Finder / explorateur de fichier et clic droit sur un dossier



Et hop dans notre terminal on est au bon endroit



Une fois à l'endroit ou l'on veut créer notre appli on va faire **npm create vite@latest** on se laisse guider par le terminal (On donne un nom au projet on sélectionne une techno (Vanilla puis Javascript))



Auteur :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

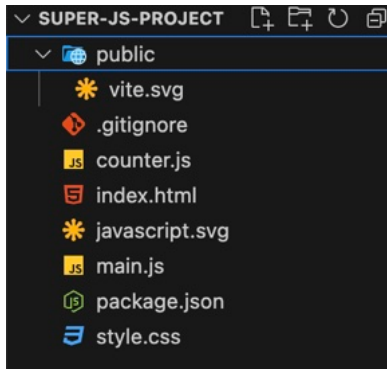
Date révision :

10/03/2023

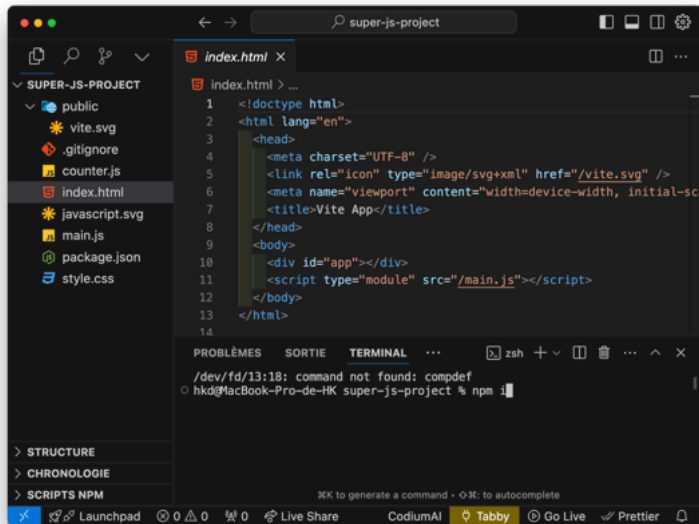


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

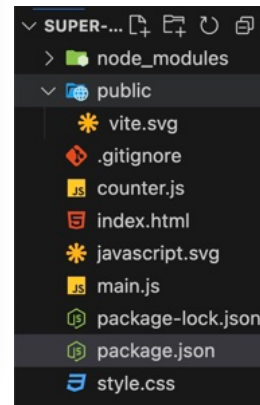
Une fois le processus terminé Vite nous a créé tout un dossier pour notre application (on peut ouvrir ce dossier dans notre IDE préféré)



Une fois dans vscode Vite propose une structure d'appli de counter de base, on peut utiliser des extensions comme live serveur dans un fichier html, mais Vite propose aussi un serveur de développement, pour l'installer on va taper dans le terminal la commande **npm i** ou **npm install**



On se retrouve alors avec un dossier node modules (un dossier où vont se télécharger les dépendances du projet (des compilateurs, un live server, etc...))



Ensuite ...

Auteur :

Jean-François Pech

Relu, validé & visé par :

Jérôme CHRETIENNE
Sophie POULAKOS
Mathieu PARIS

Date création :

03/03/2023

Date révision :

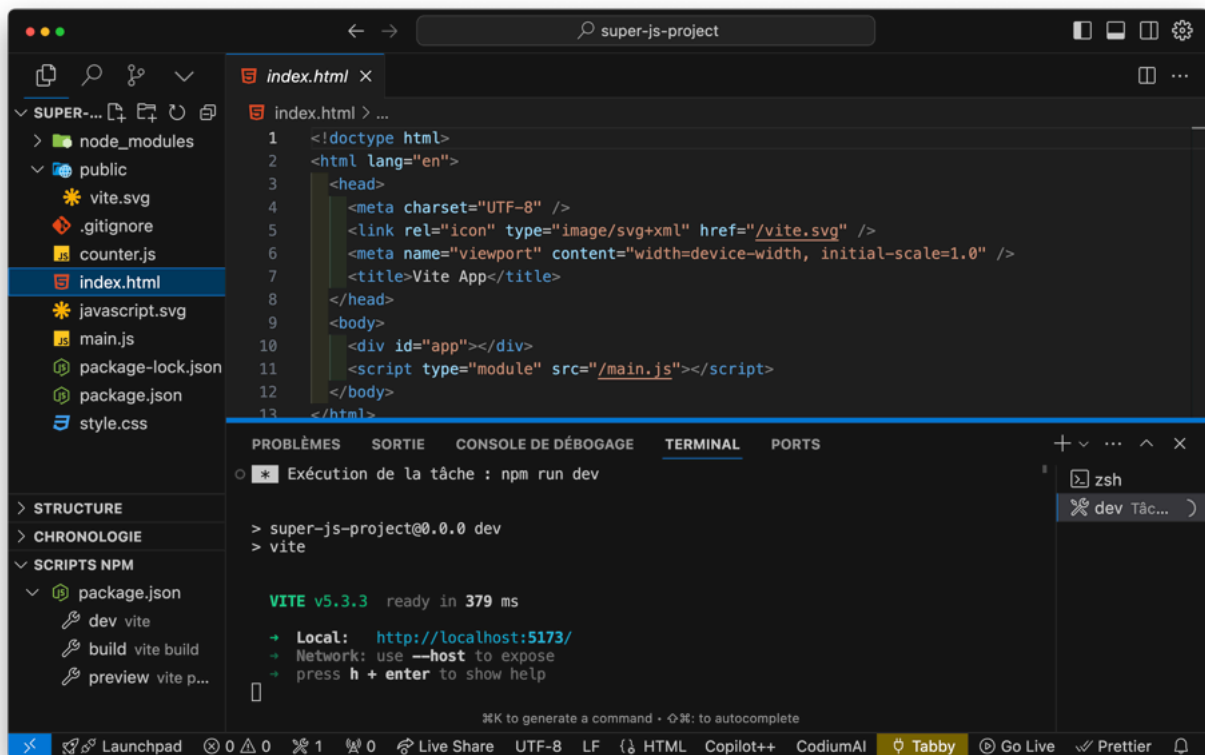
10/03/2023



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.

Toujours dans le terminal de vscode Vite nous a prévu une commande pour faire lancer l'appli sur un serveur de développement en local (équivalent de l'extension live ou five server)

Exécuter la commande **npm run dev** et hop si votre appli compile (sans erreurs) elle est dispo à cette adresse (le port change automatiquement si plusieurs app JS en même temps et il est modifiable dans un fichier de config)
<http://localhost:5173/>



Pour mieux comprendre le dossier node modules et la notion de dépendances dans un projet...

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.

Testons d'installer en local dans notre projet ... Bootstrap le framework (bibliothèque CSS), la plupart des frameworks ou bibliothèques ont à disposition des commandes npm pour faciliter l'installation.

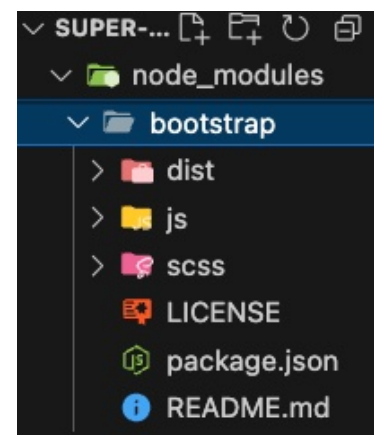
On peut ouvrir un nouveau terminal dans VS Code et taper la commande **npm i bootstrap**
(Ou **npm install bootstrap**)

```
added 2 packages, and audited 14 packages in 579ms

5 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

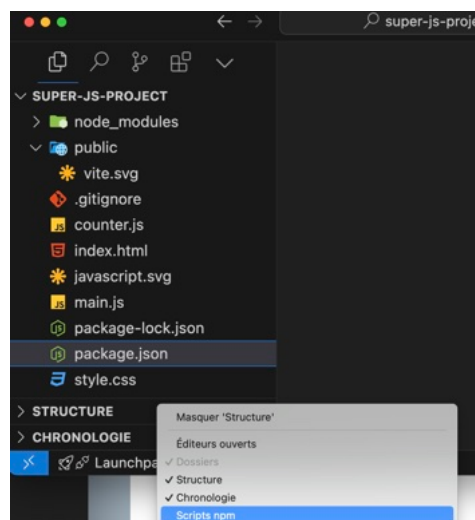
Cela va télécharger tous les fichiers du framework Bootstrap dans notre application (dans node modules)



Notre fichier package.json se met automatiquement à jour, on a une nouvelle partie dependencies avec la version de bootstrap :

```
1 {
2   "name": "super-js-project",
3   "private": true,
4   "version": "0.0.0",
5   "type": "module",
6   "scripts": {
7     "dev": "vite",
8     "build": "vite build",
9     "preview": "vite preview"
10  },
11  "devDependencies": {
12    "vite": "^5.3.1"
13  },
14  "dependencies": {
15    "bootstrap": "^5.3.3"
16  }
17 }
18
```

(Ps: vous avez aussi la partie script (quand vous faites **npm run dev**)
Astuce VS Code en bas gauche clic droit pour activer le menu script npm



Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☑ Jérôme CHRETIENNE
☑ Sophie POULAKOS
☑ Mathieu PARIS

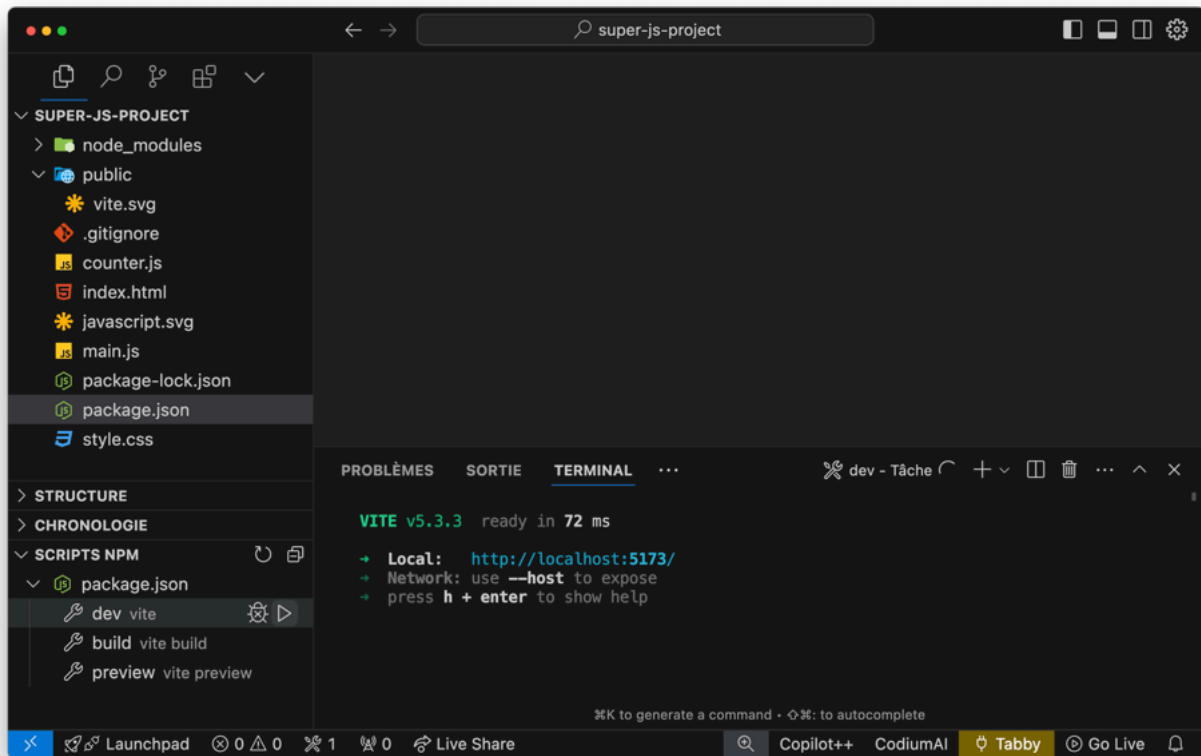
Date révision :

10/03/2023

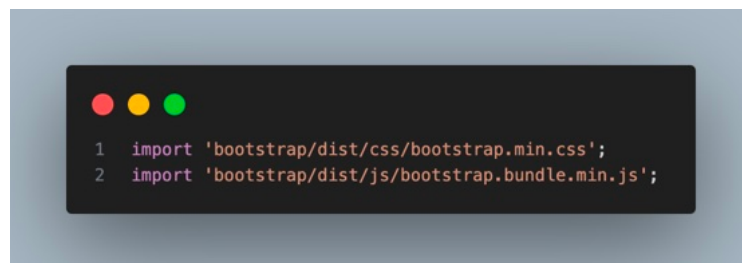


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Vite est un outil très utilisé par la plupart des frameworks basés sur JS ou TS. Cette astuce va automatiquement détecter la partie script de package.json, on a plus qu'à cliquer sur le script qu'on veut exécuter. (Tous les frameworks n'ont pas forcément les mêmes scripts, cela évite de s'embrouiller)



Maintenant on a une meilleure DX on a plus qu'à sauvegarder les fichiers et le serveur se relance automatiquement. Pour utiliser Bootstrap dans ce setup avec vite dans main.js (on va reset et garder ceci)



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
- Arrays ?

Exercice : Variables

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.

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.

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.

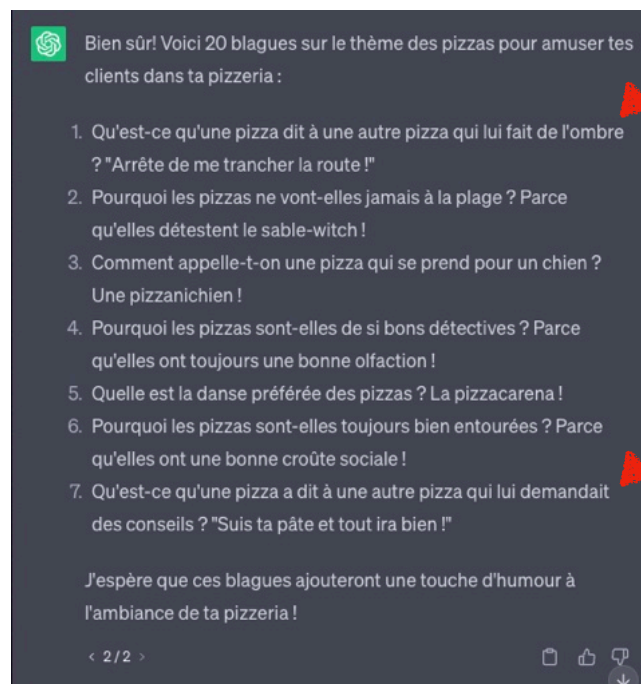
Définissez selon vous toutes les variables pertinentes qui résume la commande d'un utilisateur chez "La Pizzeria Raffinata"

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.

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.

Exercice array 2 les fonctions pour les tableaux

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

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

Exercice arrays 3

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

(🤔 Comment trouver la 1 lettre d'une chaîne de caractère (indice : comme le nom du capitaine dans Peter pan))

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

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

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);
}
```

fonctionavecParametre()

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.

Opérateurs de comparaison / condition ternaire

Opérateurs de comparaison

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

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

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

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☑ Jérôme CHRETIENNE
☑ Sophie POULAKOS
☑ Mathieu PARIS

Date révision :

10/03/2023



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

Condition / opérateur ternaire

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

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

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

On peut aussi combiner plusieurs conditions avec les opérateurs

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

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

Auteur :

Jean-François Pech

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

(version simple libre à vous d'optimiser pour avoir le meilleur algorithme)

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

Exercice : If Else

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

Auteur :

Jean-François Pech

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

```
// ? 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 :

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.

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

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

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

hasOwnProperty

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

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

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

true

app.js:31

Exercice : Objects

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

Auteur :

Jean-François Pech

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.

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

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

While

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

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

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

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

Auteur :

Jean-François Pech

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(let i=0; i<listeFilm.length; i++){  
    console.log('boucle FOR : ', listeFilm[i]);  
};
```

```
boucle FOR :   Ultime Décision      app.js:14  
boucle FOR :   Mission Alcatraz     app.js:14  
boucle FOR :   Attack Force         app.js:14
```

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.

ForEach

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

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

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

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

Auteur :

Jean-François Pech

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

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

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

For ... in

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

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

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

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

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

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

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

Auteur :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



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

Convertir des objets en tableaux

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

```
///  
// Parcourir les Objets (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 clés de l'objet converties en array : ",keyUser);  
  
const valuesUser = Object.values(userData);  
console.log("les valeurs de l'objet converties en array : ",valuesUser);  
  
const convertedDataUser = Object.entries(userData);  
console.log("les entrées de l'objet converties en array : ",convertedDataUser);
```

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

Auteur :

Jean-François Pech

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 paramètre 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 :

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.

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 :

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.

i http://127.0.0.1:5500

Les système de boucles

LOL JE SUIS HACKERMAN
LOL JE SUIS HACKERMAN
LOL JE SUIS HACKERMAN
LOL JE SUIS HACKERMAN
LOL JE SUIS HACKERMAN
LOL JE SUIS HACKERMAN
LOL JE SUIS HACKERMAN
LOL JE SUIS HACKERMAN
LOL JE SUIS HACKERMAN
LOL JE SUIS HACKERMAN
LOL JE SUIS HACKERMAN

<https://github.com/jeff404/cours-js/tree/10-boucle>

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.

Spread Operator

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

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

Utilisation avec des tableaux

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

```
const tableauOriginal = [1, 2, 3];  
const copieTableau = [...tableauOriginal];  
  
console.log(tableauOriginal);  
console.log(copieTableau);
```

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

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

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

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

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

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

Maintenant, nouvelObjet contient les mêmes propriétés que objetSource.

Auteur :

Jean-François Pech

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 operator.

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

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

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

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

Auteur :

Jean-François Pech

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 le gestion d'erreurs au travers d'exemples.

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.


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

Erreur générée par la fonction alerte

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

Erreur générée par la variable prénom

✖ ▶ Uncaught ReferenceError: prenom is not defined
 at app.js:3:1 app.js:3

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

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

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

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.



127.0.0.1:5500 says

Erreur Détectée ALERTE STOPPEZ TOUT:

Le Nom de l'erreur
ReferenceError

Le Message de l'erreur :
prenom is not defined

L'emplacement de l'erreur:
ReferenceError: prenom is not defined
at http://127.0.0.1:5500/app.js:7:5

☐ Don't allow this page to display more dialogs

OK



127.0.0.1:5500 says

Ce message s'affiche correctement

☐ Don't allow this page to display more dialogs

OK

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.

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 du corps de métier, et donc ici il nous faudra mettre en place une exception dans le cas où le montant que voudrait retiré serait supérieur au solde de l'utilisateur.

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

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

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

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

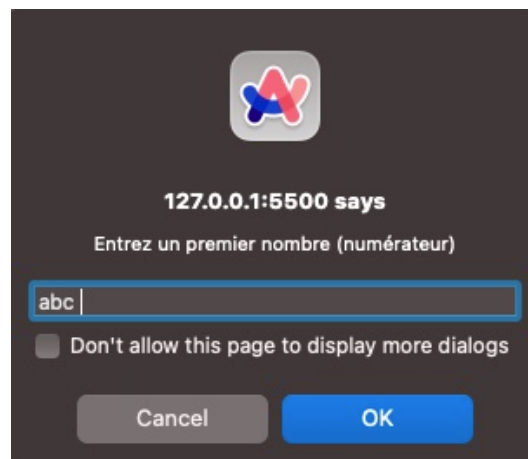
1 exception si l'utilisateur essaye de nous faire diviser par zéro

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

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

division();
```

1 ère exception :



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

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.

2^e exception :



127.0.0.1:5500 says

Entrez un deuxième nombre (dénominateur)

☐ Don't allow this page to display more dialogs

Cancel OK

✖ ▶ Uncaught Error: Division par 0 impossible app.js:31
 at div (app.js:31:15)
 at app.js:37:1

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



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

Finally

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

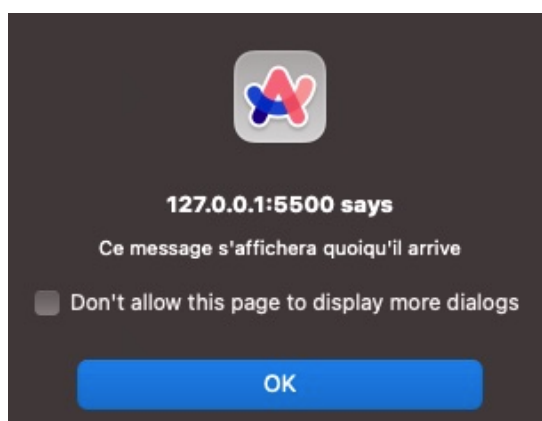
Reprenons notre fonction division : ici on ne va plus exécuter la fonction division directement dans notre programme on va **essayer** d'exécuter la fonction.

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

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

// division();

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



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.