

# Introduction

Avant tout, nous avons besoin d'un dossier de travail et d'un outil nous permettant de rédiger du code. Nous allons donc installer notre IDE : Visual Studio Code.

## A quoi sert le JS ?

Essentiellement à utiliser les ressources côté client, pas côté serveur. On peut aussi faire du serveur avec node.js. On peut faire beaucoup de choses avec le JS.

Nous allons suivre le référentiel de la certification front-end. Ainsi, dans ce cours, nous irons jusqu'à nous connecter à une API. Nous allons l'utiliser également pour vérifier les champs de formulaire.

Pour chaque cours, nous utiliserons la même structure de fichier.

1. Créer un dossier JS. Créer un dossier "1-les variables"
2. Ouvrir ce dossier avec VScode
3. Créer un index.html
4. Créer un dossier JS
5. dans le dossier JS créer un dossier script.js

## Apprendre à relier son JS à son HTML

```
<script src="./js/script.js" defer></script>
```

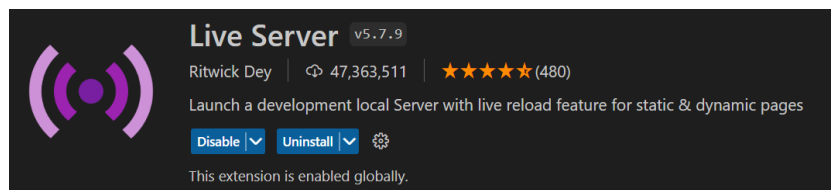
Dans le JS, ajouter un premier élément :

```
alert ('hop');
```

Ensuite, **Go live** pour valider que l'alerte se lance. Si c'est le cas, on peut considérer que le lien html/JS est OK

(si vous n'avez pas **Go live** en bas de votre IDE, il est possible que l'extension live server ne soit pas installée.

Installer Live Server dans la bibliothèque des extensions VScode



Dans le index.html, nous utiliserons le raccourci ! **tab** qui va nous préremplir les bases de notre html. Nous allons également lier le HTML et le JS en ajoutant cette ligne dans le head de notre fichier index.html

```
<script src="./js/script.js" defer></script>
```

Nous utilisons "**defer**" afin de différer le chargement du JS à la fin du chargement de la page pour rendre notre code plus performant

# 1 - Les variables

## Objectifs

- Créer une variable
- Modifier une variable
- Les opérateurs

## let

Les variables : en JS, on dit qu'on "déclare" une variable.

Pour déclarer cette variable on utilise la commande **let**

```
let a=1;
```

A cette étape, notre variable **a** à la valeur de **1**

On peut utiliser la console pour voir ce qui se passe .

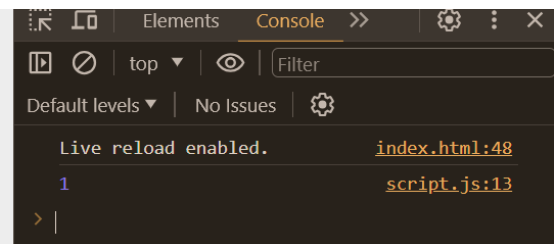
Pour afficher la variable **a** dans la console, on utilise cette commande

```
console.log (a)
```

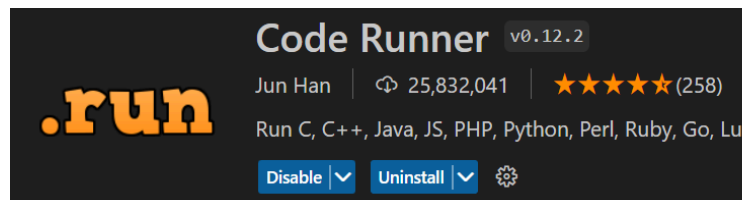
Dans le navigateur, cela se présente ainsi

img logo

## 1- Les variables



Il est aussi possible d'utiliser une extension VScode.



On a donc créé une variable.

On peut en créer une seconde : **b** et on l'affiche en console

```
let b=2;  
console.log (b)
```

## Les différents opérateurs

On peut créer des opérations :

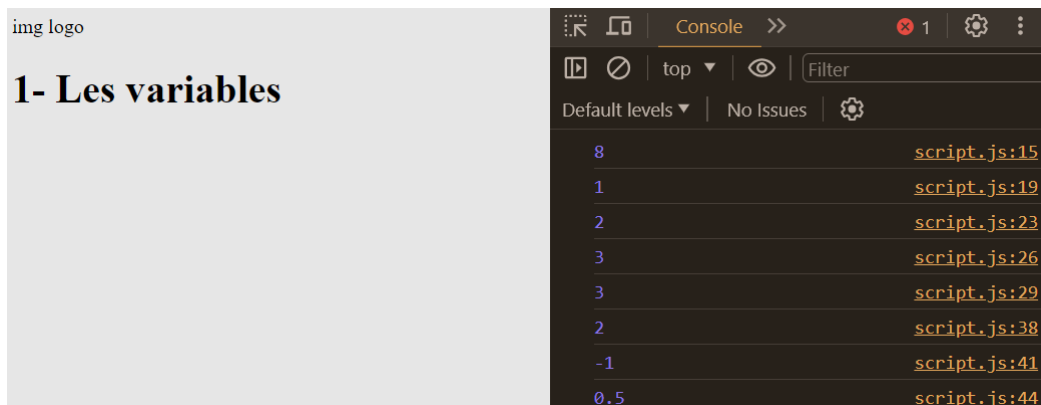
```
let resultat=a+b;  
console.log (resultat);
```

on a créé une variable "résultat" et déclarer qu'elle affichait le résultat de l'opération **a+b**

## Exercice

Avec les différents opérateurs ( \*, - , / ) faire évoluer la variable resultat  
Afficher ces résultats dans la console.

```
resultat=a*b;  
console.log (resultat)  
  
resultat=a-b;  
console.log (resultat)  
  
resultat=a/b;  
console.log (resultat)
```



## 2- les chaines de caractère

### Objectifs

- Créer une chaine de caractere
- Récupérer une chaine de caractere avec prompt
- La modifier
- L'afficher et la concatener dans la console

```
// On créé la variable prénom et nom
```

```
let prenom='Nicolas'
```

```
let nom='Le Beuzit'
```

```
// Exemple de concaténation
```

```
let message='Bonjour '+prenom;
```

```
console.log(message)
```

retour dans la console :

```
Bonjour Nicolas
```

```
// exemple de concaténation
```

```
let message='Bonjour '+prenom+' '+nom;
```

```
console.log(message)
```

retour dans la console :

```
Bonjour Nicolas Le Beuzit
```

### typeof()

La fonction **typeof()** permet de retourner le type d'une variable

```
// la fonction typeof() qui donne le type d'une variable
```

```
console.log(typeof(message));
```

retour dans la console :

```
string
```

```
script.js:26
```

On va utiliser maintenant la commande **Prompt** qui va nous permettre de récolter des informations entrées par l'utilisateur :

```
// utiliser prompt()
```

```
let prenomUtilisateur=prompt('Quel est votre prénom ?');
```

```
let nomUtilisateur=prompt('Quel est votre nom ?');
```

```
message='Bonjour '+prenomUtilisateur+' '+nomUtilisateur
```

```
console.log(message)
```

retour dans la console :

```
Bonjour michel durand
```

```
script.js:35
```

Nous allons demander au JS de s'assurer que le premier caractère est en majuscule

Pour cela, plusieurs étapes :

- Récupérer la première lettre du nom de l'utilisateur avec charat()

- On crée une variable pour stocker cette première lettre

```
let premiereLettrePrenom=prenomUtilisateur.charAt(0);
```

retour dans la console :

```
Bonjour michel durand    script.js:35
m                          script.js:46
```

## Exercice

Refaire la même opération avec le nom

```
let premiereLettreNom=nomUtilisateur.charAt(0);
console.log(premiereLettreNom);
```

retour dans la console :

```
m
d
```

## toLocaleUpperCase()

on va passer la premier lettre du prénom en majuscule avec **toLocaleUpperCase()**

```
let premiereLettrePrenomMaj=premiereLettrePrenom.toLocaleUpperCase();
console.log(premiereLettrePrenomMaj)
```

retour dans la console :

```
M
```

Idem pour le nom :

```
let premiereLettreNomMaj=premiereLettreNom.toLocaleUpperCase();
console.log(premiereLettreNomMaj)
```

retour dans la console :

```
D
```

## substring()

Maintenant que nous avons isolé la première lettre, nous avons besoin du reste de la chaîne de caractère. Pour cela, nous utilisons **substring()**. Au passage, on utilise **length** qui nous permet de remplir le second argument de **substring**.

**substring(IndexDeLaPremiereLettreAIsoler,IndexDeLaDerniereLettre)**

```
console.log(prenomUtilisateur.substring(1,prenomUtilisateur.length));
```

Grâce à cette méthode, le code nous renvoie de la seconde lettre à la dernière lettre.

retour dans la console :

```
ichel
```

Ensuite on ajoute la maj + le reste du prénom  
la maj et le reste du nom

```
console.log('Bonjour '+premiereLettrePrenomMaj+suitePrenomUtilisateur+'  
' +premiereLettreNomMaj+suiteNomUtilisateur);
```

retour dans la console :

```
Bonjour Michel Durand
```

## indexOf()

**indexOf()** est utilisé pour avoir la place d'un caractère précis dans la chaîne de caractères.  
Exemple, comment trouver le signe @ dans un mail ?

```
console.log(mail.indexOf('@'));
```

retour dans la console :

```
13
```

## Exercice

Afficher dans la console **prenomnom@nomdedomaine.com** en utilisant la concaténation, les variables et la console.

```
let mail=prenomUtilisateur+nomUtilisateur+'@'+'nomdedomaine.com'  
console.log(mail);
```

retour dans la console :

```
micheldurand@nomdedomaine.com
```

*Vous savez maintenant comment discriminer des lettres, les mettre en majuscule et concaténer le rendu.*

# 3- Les structures conditionnelles

## Objectifs

- Savoir utiliser les opérateurs de comparaison
- Savoir utiliser les conditions et / ou

```
/*je déclare trois variables */  
let a=1;  
let b=2;  
let c=1;
```

## Les 4 opérateurs principaux

- égal ==
- différent !=
- inférieur à <
- supérieur à >

```
/* l'opérateur d'égalité : == */  
if(a==b){  
    console.log('A: '+a+' est égale à B');}  
else{  
    console.log('A: '+a+', est different de B');}  
  
/*L'opérateur de non égalité : != */  
if(a!=b){  
    console.log('A: '+a+', est different de B');}  
else{  
    console.log('A: '+a+', est égale à B');}  
  
/* L'opérateur inférieur à : < */  
if(a<b){  
    console.log('A: '+a+', est inférieur à B');}  
else{  
    console.log('A: '+a+', est supérieur à B');}  
  
/* L'opérateur supérieur à : > */  
if(a>b){  
    console.log('A: '+a+', est supérieur à B');}  
else{  
    console.log('A: '+a+', est inférieur à B');}
```

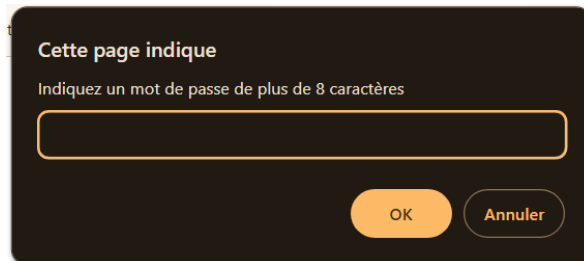
L'égalité stricte (===) , contrairement à l'égalité (==) vérifié en plus de la valeur, le type .

- comparer une chaine de caractère et un nombre dans cet exemple :

## Exercice

Créer un prompt demandant de rentrer un mot de passe d'au moins 8 caractères. Valider la longueur du mot de passe et retourner dans la console si le mot de passe est assez long, ou trop court.

```
let PasswordUser=prompt('Indiquez un mot de passe de plus de 8 caractères');
```



On crée une variable minlength de valeur 8

```
let minlength=8
```

Ici, on vérifie si le nombre de caractères du mot de passe est supérieur ou égale à la variable minlength

```
if(PasswordUser.length>=minlength) {  
    console.log('mot de passe valide')  
}  
else {  
    console.log('mot de passe le répond pas au critères demandés')  
}
```

Si la condition est remplie, on affiche “le mot de passe est valide”, sinon, “mot de passe le répond pas au critères demandés”

retour dans la console :

```
mot de passe valide
```

## L'opérateur ternaire

Il est aussi possible de créer un opérateur ternaire

```
let variable=(condition) ? 'ContenuDeLaVariableSiOK' : 'ContenuDeLaVariableSiPASOK';  
console.log(variable);
```

```
let motDePasse=(PasswordUser.length>=minlength)?'La longueur du mot de passe est  
valide':'Le mot de passe est trop court';  
console.log(motDePasse)
```



# 4- Les tableaux

## Objectifs

- créer un tableau
- accéder aux données du tableau

## Définition d'un tableaux (array)

Un tableau est comparable à une variable contenant plusieurs données.  
On accède à ces valeurs en fonction de leur position dans le tableau.

Ces tableaux sont créés comme une variable mais avec des crochets [ ] de cette manière

```
let monTableau=['force bleu','force rouge','force verte','force jaune']
```

Il est à noter qu'on compte toujours à partir de zéro. Ainsi :

- "force bleu" est à l'index 0
- "force rouge" est à l'index 1
- etc...

```
let monTableau=['force bleu','force rouge','force verte','force jaune'];  
console.log(monTableau[2]);
```

retour dans la console :

```
force verte      script.js:14
```

## push()

**push()** permet d'ajouter une valeur à la fin du tableau et on vérifie grâce à la console :

```
monTableau.push('force noire');  
console.log(monTableau);
```

retour dans la console :

```
(5) ['force bleu', 'force rouge', 'force verte', 'force jaune', 'force noire']  
  0: "force bleu"  
  1: "force rouge"  
  2: "force verte"  
  3: "force jaune"  
  4: "force noire"
```

## pop()

**pop()** permet de récupérer le **dernier élément** de notre tableau et ça le supprime du tableau. Nous allons donc supprimer l'entrée "force bleu"

```
console.log(monTableau.pop());  
console.log(monTableau);
```

retour dans la console :

```
▶ (5) ['force bleu', 'force rouge', 'force verte', 'force jaune', 'force noire'] script.js:20  
force noire script.js:22  
▶ (4) ['force bleu', 'force rouge', 'force verte', 'force jaune'] script.js:23
```

## shift()

**shift()** permet de récupérer le **premier élément** de notre tableau et ça le supprime du tableau. Nous allons donc supprimer l'entrée qu'on vient de faire. De cette manière

```
console.log(monTableau.shift());  
console.log(monTableau);
```

retour dans la console :

```
▶ (4) ['force bleu', 'force rouge', 'force verte', 'force jaune'] script.js:23  
force bleu script.js:25  
▶ (3) ['force rouge', 'force verte', 'force jaune'] script.js:26
```

## unshift()

**unshift()** permet d'ajouter une valeur **au début du tableau** et on vérifie grâce à la console :

```
monTableau.unshift('force noire');  
console.log(monTableau);
```

retour dans la console :

```
▸ (3) ['force rouge', 'force verte', 'force jaune']  
▸ (4) ['force noire', 'force rouge', 'force verte', 'force jaune']
```

## Exercice

Afficher dans la console le dernier élément de mon tableau sans le supprimer. Il faudra utiliser "length"

```
console.log(monTableau[monTableau.length-1])  
On demande d'afficher une valeur de "monTableau" avec monTableau[..]
```

Pour définir la position, nous avons besoin de sa longueur à laquelle on ajoute -1  
(car on commence à compter à partir de 0) : `monTableau[monTableau.length-1]`

# 5- Les boucles

## Objectifs

- Comprendre la logique des boucles
- Savoir utiliser les boucles **for** et **for of**

## La boucle for

Une boucle est une action qui tourne jusqu'à ce qu'une condition soit remplie.

*Un point de syntaxe avant tout :*

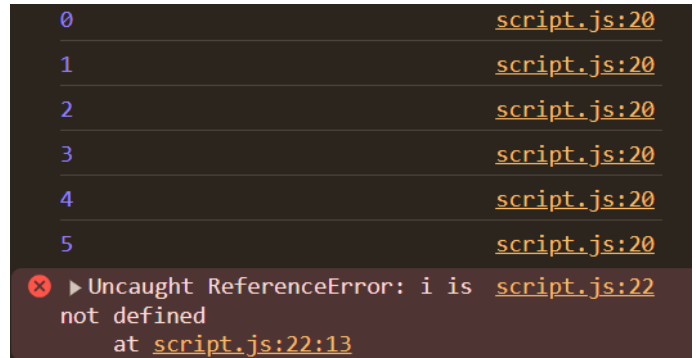
*Dans cet exemple de boucle ou on demande d'incrémenter jusqu'à 5, il est à noter que la **portée** de notre variable est circonscrite à la boucle.*

*Nous voyons clairement que la première console.log nous renvoie bien un compte de 1 jusqu'à 5.*

Si nous souhaitons afficher la variable i à l'extérieur du tableau, elle retourne une erreur.

```
for(let i=0; i<=5; i++){  
    console.log(i);  
}  
console.log(i);
```

Retour dans la console :



The screenshot shows a browser console with the following output:

0	script.js:20
1	script.js:20
2	script.js:20
3	script.js:20
4	script.js:20
5	script.js:20
✖ ▶ Uncaught ReferenceError: i is not defined at script.js:22:13	

Dans l'exemple suivant, nous allons utiliser une boucle **for** pour

La boucle for pour afficher les valeurs d'un tableau. On crée un tableau monTableau

```
let monTableau=['force rouge','force bleu','fraise',7];

for (let i=0; i<monTableau.length;i++){
  console.log(i);
  console.log(monTableau[i])
}
```

retour dans la console :

0	script.js:30
force rouge	script.js:31
1	script.js:30
force bleu	script.js:31
2	script.js:30
fraise	script.js:31
3	script.js:30
7	script.js:31

Ici, le premier console.log nous renvoie la valeur de i (0, puis 1, puis 2, etc...

Le second console.log nous renvoie la valeur contenu à cet index : 0 = force rouge, 1 = force bleu, etc...

**Exercice** (avec des éléments que nous avons vu dans “les chaînes de caractère”

*Créer un tableau nommé “ etudiant[] ” avec les étudiants suivants :  
adrien, kevin, hermine, antoine et maxime.*

*A l'aide d'une boucle, mettre chacune des premières lettres en majuscule et afficher dans la console les prénoms avec une majuscule en début de prénom.*

```
let etudiant=['adrien','kevin','hermine','antoine','maxime'];

for (
  let i=0; i<etudiant.length;i++){

/*On crée une variable qui contient la première lettre du nom de l'étudiant */
  let premiereLettrePrenom=etudiant[i].charAt(0);

/*On crée une autre variable en utilisant la méthode toLocaleUpperCase pour monter
cette lettre en majuscule */
  premiereLettrePrenomMaj=premiereLettrePrenom.toLocaleUpperCase();

/*On crée une autre variable qui contient la suite du prénom. Cette suite du prénom
est isolée grâce à la méthode substring */
  let suitePrenomUtilisateur=etudiant[i].substring(1,etudiant[i].length);
```

```
/*On fait une concaténation de la première lettre en majuscule et reste du prénom.  
Comme il est dans la boucle, ce sera répété jusqu'à la fin du nombre d'entrée dans le  
tableau */  
    console.log(premiereLettrePrenomMaj+suitePrenomUtilisateur)  
}
```

Retour dans la console :

Adrien

Kevin

Hermine

Antoine

Maxime

## Exercice :

Dans un tableau avec les étudiants suivants : adrien, hermine, kevin, antoine, maxime et lou.  
Créer une boucle qui retourne : "Bonjour adrien, hermine, antoine, maxime, lou"

```
/*Créer un premier tableau */
let etudiantTableau=['adrien','hermine','kevin','antoine','maxime','lou'];
/*Créer une variable pour débiter notre phrase : "bonjour" */
let messagebonjour='Bonjour';

/*on créé une boucle en créant une variable i qui tourne jusqu'a ce que le nombre
d'étudiant soit atteint. */
for(let i=0; i<etudiantTableau.length;i++) {
  /*Pour chaque tour de boucle, on prend la variable messagebonjour auquel on ajoute un
espace et le nom d'un étudiant i et une virgule.*/
  messagebonjour=messagebonjour+' '+etudiantTableau[i]+',';
}
console.log(messagebonjour)
```

Retour dans la console : **Bonjour adrien, hermine, kevin, antoine, maxime, lou,**

## La boucle for of

exécute un code pour chaque élément du tableau :

```
/*On créé une variable "prenom" et on lui ajoute "of".
Ce qui signifie que pour chaque élément de "etudiantTableau" on modifie le prénom à
chaque tour de boucle */

for (let prenom of etudiantTableau){
  console.log('Bonjour '+prenom)
}
```

Retour dans la console :

```
Bonjour adrien
Bonjour hermine
Bonjour kevin
Bonjour antoine
Bonjour maxime
Bonjour lou
```

## 6- Les objets

### Objectifs

- Maîtriser l’affichage et l’utilisation d’un objet
- 

Un objet se comporte un peu comme un tableau.

Ici, nous avons un stagiaire “Pierre Castrec” avec quelques **propriétés** (nom, prénom, âge et taille)

### Définir un objet

```
let stagiaire={
  prenom:'Pierre',
  nom:'Castrec',
  age: 35,
  taille: 'grand'
}
```

Chaque propriété est séparée par des virgules. Il ne faut pas ajouter de virgule à la dernière propriété.  
Pour récupérer une donnée, nous utilisons le nom de l’objet suivi de sa propriété

Dans cette exemple :

```
console.log(stagiaire.prenom)
```

Retour dans la console : **Pierre**



## Faire une liste d'objets

```
let listeDesStagiaires=[
  {
    prenom:'Pierre',
    nom:'Castrec',
    age: 35,
    taille: 'enorme'
  },
  {
    prenom:'Pierre',
    nom:'Allée',
    age: 45,
    taille: 'petit'
  },
  {
    prenom:'Nicolas',
    nom:'Le Beuzit',
    age: 42,
    taille: 'nain'
  }
]
```

*Noter l'utilisation des accolades et des crochets.*

## Exercice

Ecrire une boucle pour récupérer tous les prénoms qui sont dans le tableau “listeDesStagiaires”

Première possibilité de réponse à l'aide d'une boucle **for()** :

```
/*On crée une boucle for en définissant une variable i à 0. Si la longueur de i est
inférieure à listeDesStagiaires,
alors on ajoute 1 à i en ajoutant ++. */
for(let i=0;i<listeDesStagiaires.length;i++){

  /*Ici, on demande de retourner en console le prénom contenu dans
  listeDesStagiaires[i].prenom.
  A chaque tour, il affichera listeDesStagiaires[1].prenom, puis
  listeDesStagiaires[2].prenom ...*/
  console.log(listeDesStagiaires[i].prenom)
}
```

Seconde possibilité de réponse à l'aide d'une boucle **for of ()** .

Cette boucle est plus lisible et donc plus élégante.

```

/*Une autre manière de faire avec une boucle for of, on créé une
variable person qui évoluera à chaque élément contenu dans l'objet listeDesStagiaires
*/
for(let person of listeDesStagiaires)
{
  /*Maintenant que chaque chaque element présent dans listeDesStagiaires est identifié,
  On affiche à chaque tour le contenu en le sélectionnant ainsi : person.prenom */
  console.log(person.prenom)
  /*La boucle tournera jusqu'à la fin du tableau */
}

```

## Exercice

Dans cette liste de produits, nous voulons afficher le détail de chaque produit :

```

let inventaire = [
  {
    produit: 'Pommes',
    prix: 2.50,
    quantite: 50
  },
  {
    produit: 'Bananes',
    prix: 1.80,
    quantite: 30
  },
  {
    produit: 'Oranges',
    prix: 3.00,
    quantite: 40
  }
];

```

Une réponse possible :

```

/*On créé une boucle en créant une nouvelle variable MesProduits.
Elle évoluera à chaque tour de boucle à l'intérieur du tableau inventaire. */
for(let MesProduits of inventaire)
{
  /*A chaque tour, la boucle renverra pour chaque object de notre tableau la valeur
  MesProduits.produit, MesProduits.prix et MesProduits.quantite. */
  /*On fait un peu de concaténation pour rendre notre résultat intelligible. */
  console.log(MesProduits.produit+' au prix de '+MesProduits.prix+' €. Il nous en
  reste '+MesProduits.quantite)
};

```

Retour dans la console :

Pommes au prix de 2.5 €. Il nous en reste 50

Bananes au prix de 1.8 €. Il nous en reste 30

Oranges au prix de 3 €. Il nous en reste 40

## Exercice

Calculer le prix total du stock avec une boucle **for of**

```
/*On crée une variable total qu'on déclare à 0 qui nous sert de base pour notre
boucle */
let total=0;
/*On crée une variable prod, interne à la boucle qui sera modifié à chaque tour de
boucle et à laquelle on appliquera le code présent dans cette boucle.
*/
for(let prod of inventaire) {
/*A l'intérieur de cette boucle, on prend le total auquel on ajoute le calcul
prod.quantite*prod.prix à chaque tour, jusqu'à la fin de la liste des objets présents
dans inventaire. */
    total+=prod.quantite*prod.prix;
}

/*On renvoie dans la console la phrase suivante en intégrant le total par
concaténation */
console.log('La valeur totale de notre est stock est de '+total+' €')
```

Retour de la console :

La valeur totale de notre est stock est de 299 €

## Exercice

Une librairie mets à disposition sa liste de livre

```
let librairie = {  
  nom: "Bibliothèque municipale",  
  adresse: "123 Rue de la Bibliothèque, Ville",  
  livres: [  
    {  
      titre: "Le Seigneur des Anneaux",  
      auteur: "J.R.R. Tolkien",  
      annee_publication: 1954,  
      genre: "Fantasy",  
      disponible: true  
    },  
    {  
      titre: "Harry Potter à l'école des sorciers",  
      auteur: "J.K. Rowling",  
      annee_publication: 1997,  
      genre: "Fantasy",  
      disponible: false  
    },  
    {  
      titre: "1984",  
      auteur: "George Orwell",  
      annee_publication: 1949,  
      genre: "Dystopie",  
      disponible: true  
    }  
  ]  
};
```

Nous voulons afficher le nombre de livres disponibles dans la librairie. Il est à noter que dans nos objets figure un propriété “disponible” booléen

Réponse :

```
/*Comme précédemment on amorce notre réponse par la déclaration d'une variable
totalLivreDisponible de valeur 0. */
let totalLivreDisponible = 0

/* On créé une boucle avec une nouvelle variable livre qui évoluera à chaque tour de
boucle en fonction des objets présents dans le tableau librairie.livres */

for(let livre of librairie.livres)
{
/*Ici on mets un if : Si livre.disponible est "true" , alors on ajoute 1 à
totalLivreDisponible.*/

    if(livre.disponible) totalLivreDisponible++;
};

/*On affiche le total en console avec une concaténation pour contextualiser notre
réponse. */
console.log('Dans notre immense librairie, il reste '+totalLivreDisponible+'
livre(s)');
```

Retour de la console :

Dans notre immense librairie, il reste 2 livre(s)

# 7- Manipuler HTML

## Objectifs

- Savoir utiliser les méthodes suivantes :
  - document.querySelector
  - document.querySelectorAll
  - document.createElement()

## document.querySelector

[lien vers la documentation](#)

La méthode `querySelector()` de l'interface Document retourne le premier Element dans le document correspondant au sélecteur (ou groupe de sélecteurs) spécifié(s), ou **null** si aucune correspondance n'est trouvée.

Sélectionner et récupérer une balise HTML

```
console.log(document.querySelector('h1'));
```

retour dans la console :

```
<h1>7 - Manipuler HTML avec JS</h1>
```

Récupérer et afficher uniquement le contenu d'une balise HTML

```
console.log(document.querySelector('h1').textContent);
```

retour dans la console :

```
7 - Manipuler HTML avec JS
```

Cette utilisation nous permet de lire un contenu HTML avec le JS.

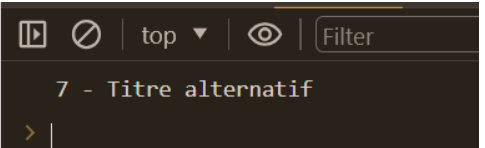
Modifier le contenu d'une balise HTML ('h1')

Pour modifier le contenu d'une balise h1, on procède ainsi :

```
document.querySelector('h1').textContent='7- Titre alternatif';  
console.log(document.querySelector('h1').textContent);
```

retour dans la console :

**7 - Titre alternatif**



Sélectionner une ID et modifier ses propriétés

```
document.querySelector('#monparagraphe').textContent='Lorem ipsum dolor sit amet. ';
```

retour dans le navigateur :

## 7 - Titre alternatif

Lorem ipsum dolor sit amet.

```
document.querySelector('h1').style.color='gray';
```

retour dans le navigateur :

## 7 - Titre alternatif

Lorem ipsum dolor sit amet.

Pour exemple, on peut également modifier des affichage (pour un menu burger par exemple)

```
document.querySelector('h1').style.display=none;
```

Pour simplifier, on peut créer une variable qui nous permettra de gagner du temps

```
let monH1=document.querySelector('h1');  
monH1.style.textDecoration='underline';  
monH1.style.fontFamily='arial';
```

retour dans le navigateur :

## 7 - Titre alternatif

Hello

Lorem ipsum dolor sit amet.



## document.querySelectorAll()

Cette méthode de Element renvoie une NodeList statique (qu'on peut comprendre comme un tableau) représentant une liste des élément du document qui correspondent au groupe de sélecteurs spécifiés

```
let mesParagraphes=document.querySelectorAll('p');
console.log(mesParagraphes);
```

retour dans la console :

```
▼ NodeList(2) [p, p#monparagraphe] ⓘ
  ▶ 0: p
  ▶ 1: p#monparagraphe
    length: 2
  ▶ [[Prototype]]: NodeList
```

On va ensuite créer une boucle **for of** pour modifier le contenu des paragraphes en leur ajoutant à chacun un peu de texte et modifiant la graisse de la typo.

```
for(let paragraphe of mesParagraphes) {
  paragraphe.textContent+=' ajout de texte à la fin du paragraphe.';
  paragraphe.style.fontWeight='bold';
}
```

retour dans le navigateur :

## 7 - Titre alternatif

Hello ajout de texte à la fin du paragraphe.

Lorem ipsum dolor sit amet. ajout de texte à la fin du paragraphe.

## document.createElement()

Comment créer un nouvel élément dans mon HTML ?

Ici, dans le HTML, nous avons créé une balise **<ul>**. Dans cette balise, nous voulons créer une balise **<li>** et la remplir avec le texte “premier point.”

Dans un premier temps, nous ajoutons une variable “myUl” qui sera la balise **<ul>** de notre document.

Dans un second temps, nous créons une balise **<li>** grâce à **document.createElement('li')**.

Ensuite, nous ajoutons le texte dans myLi avec **textContent**, puis on l’ajoute avec **myUl.appendChild(myLi)**;

```
<p>Hello</p>
<p id="monparagraphe"></p>

<ul>
</ul>
```

```
let myUl=document.querySelector('ul')
let myLi=document.createElement('li');

myLi.textContent='premier point.';
myUl.appendChild(myLi);
```

## Exercice

1. Dans un tableau **listeEtudiants=['Malcolm', 'Mo', 'Lou', 'Axel']**;

Faire une boucle pour ajouter tous les étudiants dans des balises **<li>** à l'intérieur de la balise ul **myUl**

```
let listeEtudiants=['Malcolm','Mo', 'Lou', 'Axel'];
/*Creer une variable avec le ul. */
let myUl=document.querySelector('ul');
/*Creer une boucle for of pour qu'a chaque tour on ajoute un li et on lui ajoute un
prénom de la liste */
for (let prenom of listeEtudiants){
  /*A chaque tour, on fait évoluer la variable myLiEtudiant en ajoutant une balise
li */
  let myLiEtudiant=document.createElement('li');
  /*On ajoute dans li créé la variable prénom */
  myLiEtudiant.textContent=prenom;
  /*On insere la balise li créé dans myUl déclarée plus haut. */
  myUl.appendChild(myLiEtudiant);
}
```

retour dans le navigateur :

- Malcolm
- Mo
- Lou
- Axel

## 2. Remplacer le contenu de la balise **#monparagraphe**

“Bonjour, mon nom est Mo Superman, j'ai 27 ans.”

En utilisant les données de l'**objet personne**.

```
let personne={
  prenom: 'Mo',
  nom: 'Superman',
  age: 27
}

document.querySelector('#monparagraphe').textContent='Bonjour, mon nom est '+personne.prenom+' '+personne.nom+' J\'ai '+personne.age+' ans ';
```

retour dans le navigateur :

**Bonjour, mon nom est Mo Superman J'ai 27 ans**

Une autre manière de faire des concaténation, arrivée en 2016 :

On utilise le backtick ` (alt gr + 7), puis on ajoute notre texte, les espaces, et pour appeler une variable on utilise le `${nomDeVariable}`.

```
document.querySelector('#monparagraphe').textContent=`Bonjour, mon nom est ${personne.prenom} ${personne.nom} J\'ai ${personne.age}`
```

## Serie d'exercice "superheroes"

Json :

```
const heroes = {
  "squadName" : "Super Hero Squad",
  "homeTown" : "Metro City",
  "formed" : 2016,
  "secretBase" : "Super tower",
  "active" : true,
  "members" : [
    {
      "name" : "Molecule Man",
      "age" : 29,
      "secretIdentity" : "Dan Jukes",
      "powers" : [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
    },
    {
      "name" : "Madame UpperCut",
      "age" : 39,
      "secretIdentity" : "Jane Wilson",
      "powers" : [
        "Million tonne punch",
        "Damage resistance",
        "Superhuman reflexes"
      ]
    },
    {
      "name" : "Eternal Flame",
      "age" : 1000000,
      "secretIdentity" : "Unknown",
      "powers" : [
        "Immortality",
        "Heat Immunity",
        "Inferno",
        "Teleportation",
        "Interdimensional travel"
      ]
    }
  ]
}
```

## Exercice 1

Ajoutez dans la balise header, une balise **h1** avec comme contenu la valeur de **squadName**  
Cibler ma balise **header**

```
/*On cible le Header en créant une variable myHeader */  
let myHeader=document.querySelector('header');  
/*On va créer un élément h1 */  
let myh1=document.createElement('h1');  
/*On pousse le contenu de squadName dans le h1 avec un */  
myh1.textContent=heroes.squadName;  
/*On insere la balise h1 créé dans myHeader déclarée plus haut. */  
myHeader.appendChild(myh1);
```

Retour dans le navigateur :



## Exercice 2

Toujours dans le header, ajoutez une balise p  
avec comme contenu, la valeur de hometown plus celle formed

```
/*On va créer un élément p */  
let pFormed=document.createElement('p');  
/*On pousse le contenu de demandé avec une concaténation dans le p */  
pFormed.textContent=`Basé à ${heroes.homeTown} depuis ${heroes.formed}`;  
/*On insere la balise p créé dans myHeader déclarée plus haut. */  
myHeader.appendChild(pFormed);
```

Retour dans le navigateur :



Basé à Metro City depuis 2016

### Exercice 3

Faîtes une boucle vous permettant pour chaque membres de l'équipe :

- Créer une balise **article** dans la balise **section** déjà présente
- dans cette balise **article** :
  - mettre dans **h2** le nom du **hero**
  - mettre dans une balise **p** son **identité secrete**
  - mettre dans un autre **p**, son **age**
  - mettre un 3eme **p** avec le texte "Superpouvoir"
    - Créer une **liste (li) des Superpouvoirs**

```
// je définit ma section
let mySection=document.querySelector('section');
for(let hero of heroes.members){
    // je crée la balise article
    let myArticle=document.createElement('article');

    // je crée la balise h2
    let myH2=document.createElement('h2');
    // je lui mets en contenu le nom du hero
    myH2.textContent=hero.name;

    // j'ajoute le h2 dans l'article
    myArticle.appendChild(myH2);
    // j'ajoute l'article dans la section
    mySection.appendChild(myArticle);
    // je crée un p pour l'identité secrete
    let pIdentity=document.createElement('p');
    // son contenu :
    pIdentity.textContent=hero.secretIdentity;
    // je l'ajoute dans l'article
    myArticle.appendChild(pIdentity);

    // j'ajoute un p pour l'age
    let pAge=document.createElement('p');
    // son contenu
    pAge.textContent=hero.age;
    // je l'ajoute à l'article
    myArticle.appendChild(pAge);

    // je crée un p Pouvoirs
    let pPouvoirs=document.createElement('p');
    // son contenu
    pPouvoirs.textContent='Pouvoirs :';
    // je l'ajoute à article
    myArticle.appendChild(pPouvoirs);

    // je crée une balise ul
    let myUl=document.createElement('ul');
```

```

// je l'ajoute à l'article
myArticle.appendChild(myUl);

// Ici, il faut ajouter une boucle dans la boucle. Pour chaque élément de liste
hero.powers
// on crée une variable power qui évoluera à chaque tour de boucle. Et à chaque
superpouvoir
// on ajoute un <li> et on l'ajoute dans l'<ul>

for (let power of hero.powers){
  // je crée une balise li
  let myLi=document.createElement('li');
  // son contenu
  myLi.textContent=power;
  //j'ajoute la li dans l'ul
  myUl.appendChild(myLi);
}
}

```

Retour dans le navigateur :

# SUPERHERO SQUAD

Basé à Metro City depuis 2016

## MOLECULE MAN

Dan Jukes

29

Pouvoirs :

- Radiation resistance
- Turning tiny
- Radiation blast

## MADAME UPPERCUT

Jane Wilson

39

Pouvoirs :

- Million tonne punch
- Damage resistance
- Superhuman reflexes

## ETERNAL FLAME

Unknown

1000000

Pouvoirs :

- Immortality
- Heat Immunity
- Inferno
- Teleportation
- Interdimensional travel

# 8- Les fonctions

## Objectifs

- Comprendre ce qu'est une fonction et
- Comprendre l'intérêt de la factorisation du code
- Savoir créer une fonction

## A quoi servent les fonctions ?

Une fonction sert à **factoriser** le code. A chaque fois que nous avons besoin de **créer une suite d'instructions**, il faudra prendre le réflexe de factoriser : le rendre réutilisable.

Les intérêts sont nombreux. Le fait de créer une fonction permet d'**optimiser le temps de travail**, il nous permet également de **limiter les risques de bug** et il est plus facile d'améliorer et **faire évoluer** ces blocs.

### *maPremiereFonction*

```
//D'abord on crée une fonction maPremiereFonction. A l'intérieur, on demande un
"bonjour" en console.log.
//A cette étape, elle ne s'active pas. Il faudra l'appeler pour qu'elle s'exécute
const maPremiereFonction={()=>{
  console.log('bonjour');
}}

Pour l'appeler, on procède ainsi : */
maPremiereFonction();
```

### *maSecondeFonction :*

```
// On créé une fonction avec comme paramètre "prenom".
// Dans la fonction, on envoie dans la console un "bonjour prenom" en remplissant
avec la variable créée
const maSecondeFonction=(prenom)=>{
  console.log(`Bonjour ${prenom}`)
}

// On appelle la fonction en ajoutant un prénom dans le parametre
maSecondeFonction('Pierre')
```

Retour dans la console :

Bonjour Pierre



*maTroisiemeFonction* :

Une troisième fonction dans laquelle on ajoute "prenom" en paramètre.

A l'intérieur nous créons un message "bonjour prenom", puis un "return message"

Cette commande est donc ce que la fonction "retourne". C'est le résultat de son calcul.

```
const maTroisiemeFonction=(prenom)=>{
  let message=`Bonjour ${prenom}`;
  return message;
};
// On affiche en console maTroisiemeFonction avec comme parametre Mathis.
console.log(maTroisiemeFonction('Mathis'))
```

Retour dans la console :

Bonjour Mathis

*fonctionAddition*

```
// Ma 4eme fonction. Celle-ci va nous permettre de faire une addition de 2 nombre (A
et B).
const fonctionAddition=(a,b)=>{
  let resultat=a+b;
  return resultat;
}
// On affiche dans la console le resultat d'une addition 14+34 en utilisant la
fonction :
console.log(fonctionAddition(14,34));

// On peut également ajouter un opérateur entre deux fonction par exemple :
console.log(fonctionAddition(14,34)+fonctionAddition(14,34));
```

Retour dans la console :

48

## Exercice

créer une fonction appelée "capitalize" qui prend en paramètre une chaîne de caractère et qui retourne la chaîne de caractère avec la première lettre en maj.

Une réponse possible en décomposant :

```
const capitalize=(stringCap)=>{
  //On crée une variable pour isoler la premiere lettre
  let firstLetter=stringCap.charAt(0);
  //On met cette premier lettre en majuscule
  firstLetter=firstLetter.toLocaleUpperCase();
  //On isole le reste de la chaine de caractères
  let endOfString=stringCap.substring(1,stringCap.length)
  //On concataine la premiere lettre et la seconde
  let capitalized=firstLetter+endOfString
  //On crée un return la derniere variable
  return capitalized;
}
console.log(capitalize('bonjour les gens !'));
```

Retour dans la console :

Bonjour les gens !

Une autre manière de créer cette fonction, plus élégante et plus courte :

```
const capital=(chaineDeCar)=>{
  //En créant une seule variable, on la fait évoluer ainsi
  chaineDeCar=chaineDeCar.charAt(0).toLocaleUpperCase()+chaineDeCar.substring(1);
  return chaineDeCar;
}
console.log(capital(`le jeudi c'est ravioli !`))
```

Retour dans la console :

Le jeudi c'est ravioli !

L'astuce

Il est possible d'ajouter une information @param pour préciser que chaineDeCar est une chaîne de caractère. Cela facilitera l'utilisation de l'autocomplétion de VScode : / \* \* tabulation

```
/**
 * @param {string} chaineDeCar
 * @returns
 */
```

Soit un tableau liste de personne **listePersonne=['esteban, 'samuel,'saïd, 'mathis']**

Faire une boucle pour afficher dans la console chaque prénom avec la première lettre en majuscule.

En reprenant notre fonction “capital”:

- On crée une boucle for of (créer une variable “personne” pour chaque élément de “listePersonne”
- Dans la boucle, à chaque tour de boucle, on ajoute dans la console la fonction “capital” qui s’applique à toutes les “personnes

```
let listePersonne=['esteban', 'samuel','saïd', 'mathis'];

for (let personne of listePersonne){
  console.log(capital(personne));
}
```

Retour dans la console :

Esteban

Samuel

Saïd

Mathis

## 9 - Les événements

Le principe des événements, c'est par exemple, de cliquer sur un bouton. Pour cela, nous allons utiliser des "écouteurs" d'événement **listener**

Le code se met alors à l'écoute de cet événement. Quand il arrive, cela déclenche une action.

### addEventListener()

```
<button>Bouton</button>
```

```
let myBouton=document.querySelector('button');

myBouton.addEventListener('click', ()=>{
    alert('hop');
});
```

Retour dans le navigateur :



On lui ajoute un écouteur d'événement avec **addEventListener()**.

addEventListener prend 2 paramètres : le type d'événement et une fonction à appeler  
exemple avec une fonction anonyme

```
<button id="bouton2"> Bouton2 </button>
```

```
const direBonjour=()=>{
    alert('Bonjour !');
};
// je sélectionne mon bouton2
let myBouton2=document.querySelector('#bouton2');
myBouton2.addEventListener('click', ()=>direBonjour());
```

Retour dans le navigateur :



## Exercice

Faire deux boutons : L'un qui cache une div, l'autre qui l'affiche.

Ecrivez une fonction "montrer" qui affiche la div et qui est activée par le bouton "montrer"

Ecrivez une fonction "cacher" qui cache la div et qui est activée par le bouton "cacher"

Utiliser addEventListener pour appeler ces fonctions.

```
#exoshowhide {  
  background-color: rgb(8, 59, 88) ;  
  height: 200px;  
  width: 200px;  
}
```

```
<button id="btnshow">montrer</button>  
  <button id="btnhide">cacher</button>  
  
  <div id="exoshowhide"></div>
```

```
let btnmontrer=document.querySelector('#btnshow');  
  
btnmontrer.addEventListener('click', ()=>{  
  let div=document.getElementById('exoshowhide');  
  div.style.display = 'block';  
});  
  
let btncacher=document.querySelector('#btnhide');  
  
btncacher.addEventListener('click', ()=>{  
  let div=document.getElementById('exoshowhide');  
  div.style.display = 'none';  
});
```

Retour dans le navigateur :



## Variante

1 seul bouton :

Si la div est visible, quand on click sur le bouton, ça cache div et le texte du bouton devient "Montrer"

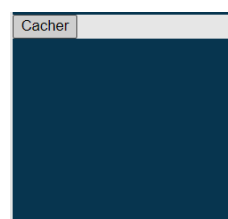
Si la div est cachée, quand on click sur le bouton, ça montre la div et le texte du bouton devient "Cacher"

1 seule fonction pour faire tout ça

```
#exoswitchdiv {  
  background-color: rgb(8, 59, 88) ;  
  height: 200px;  
  width: 200px;  
}
```

```
<button id="btnswitch">switch</button>  
<div id="exoswitchdiv"></div>
```

```
// je cible mon bouton  
let Switch=document.querySelector('#btnswitch');  
  
// je cible ma div  
let myDiv=document.getElementById('exoswitchdiv');  
  
// j'initialise une variable de type boolean (vrai ou faux)  
let toggle=true;  
  
// ma fonction  
const montrerCacher={()=>{  
  // si ma variable est vraie (la div est visible)  
  if(toggle){  
    myDiv.style.display='none';  
    Switch.textContent='Montrer';  
  }  
  else{  
    myDiv.style.display='block';  
    Switch.textContent='Cacher';  
  }  
  toggle=!toggle;  
}  
Switch.addEventListener('click', ()=>montrerCacher())
```



A remplir avec les infos du mardi 16

Ajouter les infos du premier bloc de jeudi : la géoloc



# Construire une requête dynamique

Décomposer l'url ci dessous en variable (ici, constantes, puisque nous n'allons pas les modifier)

<https://api.openweathermap.org/geo/1.0/direct?q=Dinard&limit=1&appid=43cceaf558b7f341b3ac4d5ca1175c>

On défini une constante nommée 'protocole' pour construire notre URL :

```
// On défini une constante nommée 'protocole' pour construire notre URL :
const protocole='https://';

// Le endpoint : c'est l'adresse de l'API.
const endpoint='api.openweathermap.org/';

// La route : c'est le service spécifique de l'API que nous souhaitons utiliser. CF :
// la doc de l'API.
// Le point d'interrogation sert à dire "à partir de ce symbole, on va ajouter des
// variables."
const routeGeoLoc='geo/1.0/direct?';

// On ajoute la ville dans notre url :
const place='q=Dinard';

// La limite (le nombre de résultats max souhaité)
const limit='limit=1';

// J'ajoute mon identifiant API (qu'il faut généralement souscrire payant, ou
// gratuit) 43cceaf558b7f341b3ac4d5ca1175c8c
const appid='appid=43cceaf558b7f341b3ac4d5ca1175c8c';

//Construction de la requête :
const myRequest=`${protocole}${endpoint}${routeGeoLoc}${place}&${limit}&${appid}`;

// le fetch :
fetch(myRequest)

    .then(
        response=>response.json()
    )
    .then(
        data=>console.log(data)
    )
    .catch(
        error=>console.log(error)
    )
```

Retour dans la console :  
data=>console.log(data)

```
▼ 0:
  country: "FR"
  lat: 48.6320379
  ► local_names: {br: 'Dinarzh', fr: 'Dinard', uk: 'Дінар'}
  lon: -2.0580178
  name: "Dinard"
  state: "Brittany"
```

